An Aeronautical Publish/Subscribe System Employing Imperfect Spatiotemporal Filters

Ein aeronautisches Publish/Subscribe-System mit Anwendung unvollkommener raumzeitlicher Filter

Zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation von Dipl.-Inform. (FH) Christian Grothe aus Hadamar
August 2010 — Darmstadt — D 17

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Informatik
Databases and Distributed Systems

An Aeronautical Publish/Subscribe System Employing Imperfect Spatiotemporal Filters
Ein aeronautisches Publish/Subscribe-System mit Anwendung unvollkommener raumzeitlicher Filter

Genehmigte Dissertation von Dipl.-Inform. (FH) Christian Grothe aus Hadamar

1. Gutachten: Prof. A.P. Buchmann, Ph. D.
2. Gutachten: Prof. Dr.-Ing. U. Klingauf

Tag der Einreichung: 05. Juli 2010
Tag der Prüfung: 30. August 2010

Darmstadt — D 17

Für Opa Klaus und Oma Martha, Oma Margot und Opa Bruno, die alle den Beginn dieser Arbeit miterlebt haben, aber leider keiner von ihnen den Abschluss.

Abstract

The permanent growth of air traffic volume, being a main contributor to economic growth, is reaching the limits of what current Air Traffic Management (ATM) systems are capable to handle. Fundamental changes in ATM operations are required to cope with the expected increase in the next decades.

On one hand, the bottleneck of centralized control of air space operations has been identified as a core limitation, and the two large development programmes for the future ATM systems, which are underway in Europe and North America, SESAR and NextGen, both envision a much stronger inclusion of all ATM stakeholders, from the airport operators to the individual pilot, into decision making processes, thus decentralizing ATM coordination effort. A fundamental requirement and key enabler of this relocation and sharing of control is the availability of all information necessary for making subsystem-level decisions that enhance the flow in the ATM supersystem, i.e., the decentralized availability of static and dynamic ATM data.

The traditional ATM information distribution systems on the other hand are currently undergoing a paradigm change as well, being mainly determined by a transition from paper-based, product-oriented to data-based, concept-oriented operations. The development of the *Aeronautical Information Exchange Model version 5* (AIXM 5) has pioneered this movement, enabling the Aeronautical Information domain to be the first of the ATM data domains to implement the paradigm shift. The integration of the Aeronautical Information domain with the various other ATM data domains is supported by the definition of a superordinate model system of constitutive information concepts, in which basic concepts such as space and time, are modeled in a generic, yet ATM-specific way. Such models of aeronautical spatiality and temporal dependency of aeronautical processes are presented and discussed in this thesis as AIXM 5 concepts.

Both these topics, the required reorganization of information distribution systems and the explicit modeling of ATM data domain information including generic models for basic information aspects, are in the scope of the SWIM concept: the approach to *System Wide Information Management* in the future ATM system envisioned in SESAR and NextGen. SWIM defines a high-level distributed system model for information exchange between ATM stakeholder subsystems acting as Providers or Users of information. Among the specified SWIM core communication services is publish/subscribe, a communication paradigm closely related to event-based interaction models, in which Providers can publish event notifications, and a notification service component, which inherently decouples the communicating parties, is solely responsible for mediating the notifications to affected Users who have previously expressed interest in some sort of

event notifications by registering a subscription with the service. While various subscription and event models have been proposed in the literature and have been implemented in publish/subscribe systems, none exists that provides the means to subscribe to, and efficiently disseminate, event notifiations based on spatial and temporal aspects.

In this thesis, such a subscription model for spatial and temporal aspects of event notifications for publish/subscribe systems is presented. It is derived from an initial analysis of the specific requirements of the Aeronautical Information domain with respect to spatial data and the implications from Aeronautical Information processes for a model of temporal data, and is intended to provide SWIM Users with the means to subscribe to notifications for events affecting the trajectory of a (planned or currently conducted) flight. The subscription model is based on two basic types of filters, of which subscriptions are composed and which are used in the notification service nodes for routing decisions: spatial filters and interval filters. These filter types are formally introduced, and required operations (notification matching and filter relationships) are defined. The "2.5-dimensional" aeronautical spatial model requires the adjustment of algorithms from the field of Computational Geometry to work with geometries, in which different interpolation methods for lines are used, accounting for different types of flight paths common in air navigation, namely loxodrome (rhumb line) and orthodrome (great circle track) routes.

Past research in distributed content-based publish/subscribe systems has yielded advanced routing algorithms that exploit equality and inclusion relations of subscriptions to reduce filter handling overhead in the distributed nodes of the notification service. In the given context though, these traditional approaches are uneffective due to lack of appropriate subscription relationships because the 4D trajectories of two different flights never assume the same spacetime nor is one trajectory's spacetime covered by another trajectory's one. Nevertheless, their spacetime may be close enough such that the flights are affected by the same events, in which case the respective subscriptions could be united, thus reducing filter handling overhead throughout the distributed nodes of the system. The main challenge of such an approach of merging similar subscriptions is to decide if filters are "similar enough" to be merged. The problem of (imperfectly) merging filters is based on a trade-off between filtering quality and filter numbers, and a formal approach to describing filter similarity and filtering quality is required to sensibly balance this trade-off. Such a formal discussion is presented in this thesis, and heuristical algorithms are proposed that integrate with merging-based routing algorithms. The proposed approaches are based on quantitatively estimating the reduction of filtering quality involved in creating a filter merger. This merger quality estimation takes into account simple filter metrics (size and distance) defined for this purpose, and returns the estimation as numeric value. By defining a merger quality

threshold value, the trade-off of reduction (of the number of filters) against precision (filtering quality) can be easily adjusted as required by the application by means of a single numeric value.

The proposed size-based and distance-based approaches to merger quality estimation were evaluated in filter merging experiments, and the results show that an approach based on normalized filter distance outperforms the others with respect to achieved filtering quality. Furthermore, the impact of other factors such as filter and notification size, distance and distribution characteristics (clustered or uniformly distributed) as well as filter and notification numbers was examined, and it is found that higher filter and notification density in the filter space, which is the representation of similar interests of Users and event hotspots, generally allows for better filtering quality when applying imperfect filter merging.

Algorithms for the filter handling operations of a broker node in a distributed publish/subscribe system are presented that employ filter merging aiming to increase system scalability by reducing routing table size and filter forwarding overhead at the cost of unnecessarily forwarded notifications that result from the reduction of filtering quality. The scalability benefits of the approach are evaluated, firstly, by formally describing the effects of the presented filter handling scheme and investigating the propagation of the effects throughout the broker network, and secondly, in experiments using realistic flight subscriptions derived by extrapolating real historic flight data. The analysis of the results shows that very specific conditions have to be met for the overall approach to reduce subscription forwarding overhead more than increase notification forwarding overhead compared to a simple routing approach. Nevertheless, statements are derived on the conditions for filter and notification characteristics that must be met for the approaches to be effective.

Zusammenfassung

Das permanente Wachstum des Luftverkehrsaufkommens, das einen wesentlichen Beitrag zu wirtschaftlichem Wachstum liefert, erreicht die Grenze dessen, was aktuelle Air Traffic Management (ATM) Systeme bewältigen können. Grundlegende Änderungen im ATM-Betrieb sind notwendig, um die für die nächsten Jahrzehnte erwartete Steigerung bewältigen zu können.

Auf der einen Seite wurde der Flaschenhals der zentralisierten Steuerung von Luftraumbetrieb als eine essenzielle Beschränkung identifiziert, und die beiden großen Entwicklungsprogramme, die in Europa und Nordamerika im Gange sind, SESAR und NextGen, sehen beide eine viel stärkere Einbindung aller ATM-Beteiligten in Entscheidungsfindungsprozesse vor, um so den Aufwand für ATM-Koordination zu dezentralisieren. Eine wesentliche Anforderung und ein Treiber für diese Steuerungsverlagerung und -verteilung ist die Verfügbarkeit aller Informationen für auf Subsystem-Ebene zu treffenden Entscheidungen, die den Fluss im ATM-Supersystem verbessern, also die dezentrale Verfügbarkeit von statischen und dynamischen ATM-Daten.

Auf der anderen Seite findet in den traditionellen ATM-Informationsverteilungssystemen gerade ein Paradigmenwechsel statt, der im Wesentlichen bestimmt wird durch einen Übergang von Papier-basierten, Produkt-orientierten Abläufen zu Daten-basierten, Konzept-orientierten. Die Entwicklung des „Aeronautical Information Exchange Model version 5“ (AIXM 5) hat dieser Bewegung den Weg bereitet und es damit dem Bereich der aeronautischen Informationen ermöglicht, die erste ATM-Daten-Domäne zu sein, die den Paradigmenwechsel umsetzt. Die Integration der Domäne aeronautischer Informationen mit den verschiedenen anderen ATM-Daten-Domänen wird unterstützt durch die Definition eines übergeordneten Modellsystems grundlegender Informationskonzepte, dem „ATM Information Reference model“. In diesem sind elementare Konzepte wie Raum und Zeit generisch, aber dennoch ATM-spezifisch modelliert. Solche Modelle aeronautischer Räumlichkeit und zeitlicher Abhängigkeit aeronautischer Prozesse werden in dieser Arbeit als Konzepte von AIXM 5 vorgestellt und diskutiert.

Beide Themen, die benötigte Reorganisation der Informationsverteilungssysteme und die explizite Modellierung von Information in ATM-Daten-Domänen unter Einbeziehung generischer Modelle für elementare Informationsaspekte, sind Bestandteile des „SWIM“-Konzepts: der Ansatz zu *System Wide Information Management* in den von SESAR und NextGen vorgesehenen zukünftigen ATM-Systemen. SWIM definiert ein abstraktes Modell eines verteilten Systems für Informationsaustausch zwischen den Subsystemen der ATM-Beteiligten, die als „Provider“ und „User“ von Information agieren. Einer

der fundamentalen SWIM-Kommunikationsdienste ist Publish/Subscribe (Pub/Sub): ein Kommunikationsparadigma, das in enger Beziehung zu ereignisbasierten Interaktionsmodellen steht. In Pub/Sub können Provider Ereignismeldungen (*event notifications*) publizieren. Ein Benachrichtigungsdienst, der die Kommunikationsparteien inhärent entkoppelt, ist alleine verantwortlich für die Vermittlung der Meldungen an betroffene User, die vorher ihr Interesse an einer bestimmten Art von Ereignismeldungen ausgedrückt haben, indem sie bei dem Dienst eine Subskription angemeldet haben. Obwohl diverse Subskriptions- und Ereignismodelle in der Literatur vorgeschlagen und in Pub/Sub-Systemen implementiert worden sind, gibt es keines, das die Mittel bereitstellt, Subskriptionen auf der Basis von räumlichen und zeitlichen Aspekten von Ereignissen auszudrücken, und auf der selben Grundlage Ereignismeldungen effizient zu verteilen.

In dieser Arbeit wird solch ein Subskriptionsmodell für räumliche und zeitliche Aspekte von Ereignismeldungen vorgestellt. Es leitet sich ab aus einer initialen Analyse der speziellen Anforderungen der Domäne aeronautischer Informationen bezüglich räumlicher Daten und den Implikationen aus den Abläufen in dieser Domäne für ein Modell temporaler Daten und verfolgt das Ziel, die Mittel zur Verfügung zu stellen, sich für Meldungen über Ereignisse zu subskribieren, die sich auf die Trajektorie eines (geplanten oder in Durchführung befindlichen) Fluges auswirken. Das Subskriptionsmodell basiert auf zwei elementaren Arten von Filtern, aus denen Subskriptionen zusammengesetzt werden und die im Benachrichtigungsdienst für Routing-Entscheidungen benutzt werden: räumliche Filter und Intervall-Filter. Diese Filtertypen werden formal eingeführt, und benötigte Operationen werden definiert, nämlich das Anwenden von Filtern auf Meldungen (*notification matching*) und Beziehungen zwischen Filtern. Das „2,5-dimensionale“ aeronautische Raum-Modell erfordert die Anpassung von Algorithmen aus dem Bereich der Algorithmischen Geometrie, um auf Geometrien angewendet werden zu können, bei denen unterschiedliche Interpolationsverfahren für Linien zur Anwendung kommen. Dies ist notwendig aufgrund der verschiedenen Arten von in der Flugnavigation gebräuchlichen Pfaden, nämlich Loxodrome- (*rhumb line*) und Orthodrome-(Großkreis-)Routen.

Forschung an verteilten inhaltsbasierten Pub/Sub-Systemen hat in der Vergangenheit fortgeschrittene Routing-Algorithmen hervorgebracht, die Gleichheits- und Überdeckungsbeziehungen zwischen Subskriptionen ausnutzen, um den Mehraufwand für den Umgang mit Filtern (*filter handling overhead*) zu reduzieren, der in den verteilten Knoten des Benachrichtigungsdienstes anfällt. Im hier gegebenen Kontext allerdings sind diese traditionellen Ansätze wirkungslos, da die Subskriptionen die entsprechenden Beziehungen nicht aufweisen, denn die 4D-Trajektorien zweier Flüge nehmen niemals die gleiche Raumzeit ein. Ebenso kann die Raumzeit einer der Trajektorien nie von der der anderen überdeckt sein. Nichtsdestotrotz können die jeweiligen Raumzeiten

nahe genug beieinander sein, so dass die Flüge von den gleichen Ereignissen betroffen sind. In diesem Fall könnten die jeweiligen Subskriptionen vereinigt werden, um so den *filter handling overhead* über alle verteilten Knoten des Systems zu reduzieren. Die zentrale Herausforderung solch eines Ansatzes zum Zusammenfügen (*merging*) ähnlicher Subskriptionen ist die Entscheidung, ob Filter „ähnlich genug“ sind, um zusammengefügt zu werden. Dem Problem des unvollkommenen Zusammenfügens (*imperfect merging*) von Filtern liegt ein Trade-off zugrunde, nämlich zwischen Filterungsqualität und der Anzahl von Filtern. Ein formaler Ansatz zur Beschreibung von Filter-Ähnlichkeit und Filterungsqualität ist notwendig, um diesen Trade-off sinnvoll auszutarieren. Solch eine formale Diskussion wird in dieser Arbeit präsentiert, und heuristische Algorithmen werden vorgeschlagen, die in auf *filter merging* basierenden Routing-Algorithmen eingesetzt werden können. Die vorgeschlagenen Verfahren gründen darauf, die Abnahme der Filterungsqualität durch *filter merging* quantitativ abzuschätzen. Die Schätzung der Qualität eines zusammengefügten Filters (*filter merger*) basiert auf einfachen, für diesen Zweck definierten Filter-Metriken (Größe und Distanz) und liefert als Ergebnis die Schätzung als Zahlenwert. Indem ein Grenzwert für die Qualität von *filter mergers* definiert wird, kann der Trade-off zwischen „Reduktion“ (der Filter-Anzahl) und „Präzision“ (Filterungsqualität) einfach nach den Anforderungen der Anwendung anhand eines einfachen Zahlenwerts ausgeregelt werden.

Die vorgeschlagenen Größe- und Distanz-basierten Ansätze zur Qualitätsschätzung von *filter mergers* wurden in Experimenten getestet, deren Ergebnisse zeigen, dass ein auf normalisierter Filterdistanz basierender Ansatz den anderen im Hinblick auf erreichte Filterungsqualität überlegen ist. Darüber hinaus wurde der Einfluss anderer Faktoren untersucht, wie der Größe von Filtern und Ereignismeldungen, ihrer Distanz und der Verteilungscharakteristik (gehäuft/*clustered* oder gleichverteilt) sowie der Anzahl von Filtern und Ereignismeldungen. Es zeigt sich, dass eine höhere Dichte von Filtern und Ereignismeldungen im Filterraum (welche eine Abbildung ähnlicher Interessen der User und von Ereignis-Hotspots darstellt) generell eine höhere Filterungsqualität ermöglicht, wenn *imperfect merging* angewendet wird.

Weiterhin werden in dieser Arbeit Algorithmen für die Filter-Verarbeitung eines Vermittler-Knotens in einem verteilten Pub/Sub-System vorgestellt, die *filter merging* mit dem Ziel anwenden, die Skalierbarkeit des Systems zu erhöhen, indem die Größe der Routing-Tabellen und der Mehraufwand für das Weiterleiten von Filtern reduziert wird unter Inkaufnahme von unnötigerweise weitergeleiteten Ereignismeldungen durch die Abnahme der Filterungsqualität. Die Vorteile für die Skalierbarkeit des Ansatzes werden in dieser Arbeit auf zwei Arten untersucht und beurteilt: zunächst durch die formale Beschreibung der Auswirkungen des vorgestellten Schemas zur Filter-Verarbeitung und die Untersuchung der Fortpflanzung der Auswirkungen durch das

Vermittler-Netzwerk, dann in Experimenten mit realistischen Flug-Subskriptionen, die durch Extrapolation von Daten realer, vergangener Flüge erzeugt wurden. Die Analyse der Ergebnisse zeigt, dass sehr spezifische Bedingungen erfüllt sein müssen, damit der Gesamtansatz die Anzahl von Filter-Operationen (Verarbeitung und Weiterleitung) stärker reduziert als er die Anzahl entsprechender Ereignismeldung-Operationen erhöht. Nichtsdestotrotz gelingt es, Aussagen aus den Ergebnissen über die Bedingungen abzuleiten, die erfüllt sein müssen, damit das Verfahren erfolgreich ist.

Preface

Acknowledgments

First of all, I would like to thank my advisor Prof. Alejandro Buchmann, Ph.D., and my supervisor Prof. Dr.-Ing. Uwe Klingauf, for their help and advice over the last years, and for assuming the tasks of *Referent* and *Korreferent*. Special thanks also go to Tsvetan Penev for being a good colleague and challenging discussion partner and for contributing some question marks to my work, my other colleagues at the Aviation Databases group in particular and at the *Fachgebiet Flugsysteme und Regelungstechnik (FSR)* and the *Fachgebiet Datenbanken und Verteilte Systeme (DVS)* in general for the good cooperation and good time I had at the Technische Universität Darmstadt, and my international colleagues from EUROCONTROL, the FAA, and other institutions, for the inspiring discussions we had at and after conferences, congresses, and workshops.

Last but not least, I thank my family. Without the continuous backing of my parents and the confidence and strength to bear one or another setback that are based on supporting family ties, this work would not have been possible.

Inputs to this Thesis

Results and inputs from various areas come together in this thesis. While being employed as Wissenschaftlicher Mitarbeiter at FSR and affiliated with DVS, I had the chance to take part in various research and development activities in the aviation domain.

The work on a thesis titled “Updates für Onboard-Datenbanken” [74] in 2005 created an initial interest in AIXM. After initiating a cooperation between EUROCONTROL and FSR, I was involved in the drafting group of AIXM 5 from the beginning in 2005, and the development and implementation of the AIXM 5 Temporality Model has been one of the main topics of research and interest in the past five years. Two studies for the verification and validation as well as prototypical implementation of the Temporality Model were conducted in 2005/6 and 2006/7. The results and findings of the studies as well as different aspects of AIXM 5 and its Temporality Model were published as official documentation by EUROCONTROL [9, 68, 77, 79] and presented at various congresses and conferences [75, 76, 78, 80, 81, 82] and published in the proceedings [85, 83, 86]. Like every other research topic in the aviation domain within the last years, these activities must be seen in the larger frame of SESAR and NextGen (and preceding programmes)

and the SWIM approach. Studying the details and various aspects of these programmes has therefore always been part of the work on those topics.

The investigation of publish/subscribe and event-based systems always accompanied the aviation-focused activities. These topics have been in the focus of research at DVS for several years and interesting discussions at, and other publications out of, DVS layed the foundations for the work presented here. The supervision of a master's thesis in 2007 [146], made possible by a cooperation between FSR and DVS, was an additional building block, and REBECA, the implementation of a pub/sub system with a distributed notification service many members of DVS have contributed to, has been used as the architectural framework and implementation base of this work.

This thesis finally joins all results and findings of the aforementioned activities by proposing a spatiotemporal filter and content model based on AIXM 5's spatial and temporal model for a distributed event notification system that can serve as a SWIM core service. The filter model, the formalization of spatial and interval filters, the approach to filter merging and the development of respective heuristics are published here for the first time. Other publications [87, 88] are also related to, but not incorporated in, this thesis.

Contents

List of Figures

List of Abbreviations and Acronyms

ACARE	Advisory Council for Aeronautics Research in Europe
AIM	Aeronautical Information Management
AIP	Aeronautical Information Publication
AIRAC	Aeronautical Information Regulation And Control
AIS	Aeronautical Information Services
AIXM	Aeronautical Information eXchange Model
AIXM 5	Aeronautical Information eXchange Model version 5
AMDT	Amendment (to an AIP)
ANSP	Air Navigation Service Provider
ATC	Air Traffic Control
ATM	Air Traffic Management
BDT	Business Development Trajectory
BT	Business Trajectory
BREP	boundary representation
CAD	Computer Aided Design
CDM	Collaborative Decision Making
CFMU	Central Flow Management Unit
CG	Computational Geometry
PIP	point-in-polygon problem
CM	Conceptual Model
DCEL	Doubly-Connected Edge List
DEBS	Distributed Event-Based System
DHT	Distributed Hash Table

DME Distance Measuring Equipment

DVS Fachgebiet Datenbanken und Verteilte Systeme

EC European Commission

EGM-96 Earth Gravitational Model 1996

EUROCONTROL European Organisation for the Safety of Air Navigation

FAA Federal Aviation Administration

FL flight level

FMS Flight Management System

FSR Fachgebiet Flugsysteme und Regelungstechnik

GML Geography Markup Language

GNSS Global Navigation Satellite System

GIS Geographic Information System

GPS Global Positioning System

ICAO International Civil Aviation Organization

IFR Instrument Flight Rules

ISO International Organization for Standardization

MBR Minimum Bounding Rectangle

MSL mean sea level

NAS National Airspace System

NASA National Aeronautics and Space Administration

NAT North Atlantic Track

NextGen Next Generation Air Transport System

nm nautical mile

NOTAM Notice to Airmen

OGC Open Geospatial Consortium

OMG Object Management Group

P2P	peer-to-peer
RBT	Reference Business Trajectory
SBT	Shared Business Trajectory
SESAR	Single European Sky ATM Research Initiative
SFS	Simple Feature Specification
SWIM	System Wide Information Management
UIR	Upper Flight Information Region
UML	Unified Modeling Language
VOR	VHF Omnidirectional Range
WGS-84	World Geodetic System 1984
XML	eXtensible Markup Language

List of Symbols

F, G	filters
$\lvert F \rvert$	size of (interval filter) F
$\tilde{f}$	centroid of (interval filter) F
$F \equiv G$	F and G are equal
$F \sqcap G$	F and G intersect
$F \not\sqcap G$	F and G are disjoint
$F \sqsupseteq G$	F covers G
$F \sqsupset G$	F properly covers G
$F \sqcup G$	F is merged with G
$d(F, G)$	distance of (interval filters) F and G
m, n	notifications
$\lvert n \rvert$	size of (interval notification) n
$d(m, n)$	distance of (interval notifications) m and n
$F(n)$	F matches n
$N(F)$	set of all notifications matched by F
$\mathscr{F}, \mathscr{G}$	filter sets
$p(\mathscr{F})$	precision of $\mathscr{F}$
$r(\mathscr{F})$	reduction of $\mathscr{F}$
$c(\mathscr{F})$	cover probability of $\mathscr{F}$
$m(\mathscr{F})$	matching probability of $\mathscr{F}$
$\overline{d}(\mathscr{F})$	mean distance of filters in $\mathscr{F}$
$\mathbb{F}$	set of all filters
$\mathscr{N}$	notification set
$\overline{d}(\mathscr{N})$	mean distance of filters in $\mathscr{N}$
$\mathbb{N}$	set of all notifications (or natural numbers)
B, B_x	brokers
$\mathscr{D}$	set of all neighbors (of a broker)
$m(F)$	set of constituent filters of (merger filter) F
$\mathscr{R}_x$	set of filters in entries of routing table of broker B_x
$\mathscr{R}_x^{\rightarrow y}$	set of filters in $\mathscr{R}_x$ towards broker B_y
$O_\mathrm{n}\left(B_x \rightarrow B_y\right)$	notification forwarding overhead (increase) from B_x to B_y
$O_\mathrm{f}\left(B_x \rightarrow B_y\right)$	filter forwarding overhead reduction from B_x to B_y

1 Introduction: Information Distribution in the Aviation Domain

In any networked information exchange application, the goal is to make available the *right information* at the *right place* at the *right time*. In the Air Traffic Management (ATM) world though, this is a vital requirement for flight safety and thus for the lives of thousands of people at any moment in time on one hand and for the efficiency of one of the primary enabler for economic growth, the aviation industry, on the other hand.

During the flight planning phase as well as for conduction and surveillance of a flight, a large amount of data has to be exchanged in a timely fashion between the airline's flight planning and operational control departments, the aircraft itself, air traffic control, the airport operator, and other stakeholders, such as airspace environment planning data like airway availability and airspace restrictions, airport operation information like runway closures and approach procedure effectivity and meteorological conditions like weather reports and forecasts.

The inherent scalability limitations of the currently installed information distribution system, however, are seen as a major impediment to the future ATM and Air Transport system that is required to meet the challenge of constantly increasing air traffic. The global distribution of information sources and its users, the diverse requirements of the users, and the wide heterogeneity of the distribution system's parts already pose big challenges to efficient information distribution. In addition, we are currently facing a situation in this environment where information is produced and published by local, regional, national and multinational authorities all over the world, often in mainly unstructured, clear-text, paper-based products, and sent through various channels and over different networks and mediators to eventually reach all concerned people and institutions, which are also distributed around the globe. A huge effort is being made by all stakeholders to assure to a certain degree that all aviation personnel receives all the information that concern their task, be it the planning, conduction or surveillance of a flight, or the manufacturing of products like aeronautical charts and electronic onboard applications.

This issue is among others in the scope of large aviation system modernization programmes currently put in place in Europe and the United States, namely the *Single European Sky ATM Research Initiative (*SESAR*)* and the *Next Generation Air Transport System (NextGen)*. The common vision of future information exchange in these programs is to get rid of the dedicated communication links and historic formats used for information distribution that currently exist between the different aviation stakeholders and to establish a concept called *System Wide Information Management (*SWIM*)*, in which

the primary focus is on the development and application of standards for data models, communication services, and processes.

This introductory chapter first reviews in Section 1.1 the current situation and the holistic approach of ATM and airspace system modernization taken in SESAR and NextGen. SWIM is described in more detail in Section 1.2 together with its demand of a publish/subscribe core communication service. Section 1.3 brings the topics together and narrows down the scope of this thesis, also presenting the specific contributions, and the organization of the remainder of this work.

1.1 The Future ATM System Envisioned in SESAR and NextGen

Worldwide air traffic has experienced a continuous growth in the last decades, and in recent years, this trend has even accelerated. The total number of flights in managed airspace over Europe increased by 5.3 % from 2006 to 2007, 2007 being the first year ever with more than 10 million recorded Instrument Flight Rules (IFR) movements, with a record number of 33,506 flights on 31 August 2007. So far, optimized procedures and processes in the ATM system have been able to cope with this increase, keeping flight safety and timeliness at a nearly constant level [69].

While 2008 is the first year in decades that has seen a stagnation due to the economic crisis, the 2008 long-term forecast by the European Organisation for the Safety of Air Navigation EUROCONTROL [64] suggests that there will be between 16.5 and 22.1 million IFR flights in Europe by 2030, which is between 1.7 and 2.2 times more than in 2007 [66]. Coping with the increase in air transportation demand and at the same time reducing aircraft's impact on the environment, improving safety, efficiency, and timeliness are the pronounced goals of ATM research, declared in 2001 in the seminal "Vision 2020" of the *Advisory Council for Aeronautics Research in Europe (ACARE)* [3], in the *Strategic Research Agenda* in 2001 and 2004 [4, 5], and the 2008 addendum thereto taking into account the changed circumstances [6].

The challenges to meet these goals are manyfold, and in essence it has been found that fundamentally new approaches have to be taken for various aspects of ATM to prepare for the future. A "paradigm shift" is necessary [117]. Among other areas, the complete redesign of current information handling processes and communication systems is believed to be a crucial prerequisite for an efficiency increase in ATM [158, 112, 59]. To prepare for these challenges, the European Commission (EC) together with EUROCONTROL has set up the European ATM modernisation programme SESAR. It is expected to "deliver a future European ATM System for 2020 and beyond which can, relative to today's performance, enable up to a 3-fold increase in air traffic movements whilst reducing delays, improve the safety performance by a factor of 10, enable a 10 %

reduction in the effects aircraft have on the environment and provide ATM services at a cost to the airspace users which is at least 50 % less." [156]

SESAR started in 2005. As of beginning of 2009, the first of its three major phases (Definition) has just concluded and the Development phase is starting. The final Deployment phase is planned to start in 2014 (Figure 1.1). SESAR is expected to cost more than 22 Billion Euros, which are funded partly by the EC and EUROCONTROL (in the first two phases) [65].

Figure 1.1.: SESAR Major Phases (from [65]).

Air traffic industry's challenges and expectations are similar in North America, where the Federal Aviation Administration (FAA) has also set up a research and development programme for the National Airspace System (NAS), intended to define processes and technologies for the Next Generation Air Transport System (NextGen) [111, 112].

It has become common sense with the programmes' stakeholders that the envisioned improvements cannot be achieved by merely enhancing the current systems and processes, but only a fundamental rethinking of traditional processes and in some parts a complete redesign of the ATM system will allow for meeting the future challenges.

The central concepts and conceptual building blocks of the future ATM system envisioned in both, SESAR and NextGen, are *Collaborative Decision Making (CDM)*, the *4D Business Trajectory (BT)*, and SWIM [158, 112, 166, 130, 48, 172].

1.1.1 Collaborative Decision Making and the Business Trajectory

Air Traffic Management begins in a planning phase before the actual execution of a flight. In Europe today, EUROCONTROL's *Central Flow Management Unit (CFMU)*, which is responsible for the Europe-wide assignment of time slots for approach and take-off at high-density airports, prearranges planned flights based on filed flight plans and available capacity in airspace and at airports in coordination with Air Navigation Service Providers (ANSPs) supplying Air Traffic Control (ATC) services at the national level.

During the execution of a flight, the surveillance and tactical management of air traffic is the sole responsibility of ATC centers, who monitor the aircraft and assign flight levels and speeds individually in communication with the pilots.

This centralized management of air traffic is an intrinsic impediment to airspace capacity increase and hence overall system scalability. A central component of the future ATM system is therefore an increased involvement of all stakeholders in decision making, which is referred to as *Collaborative Decision Making* [166].

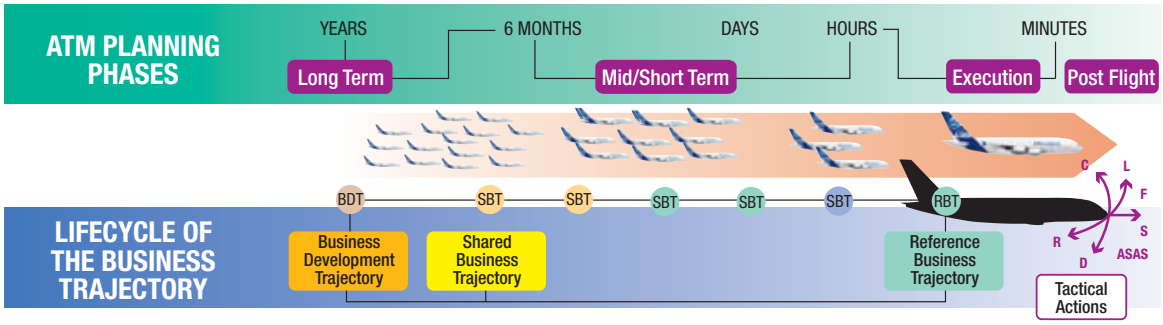


Figure 1.2.: Lifecycle of the Business Trajectory involving Collaborative Decision Making (from [158]).

The CDM process for a flight starts months before the flight conduction with the development of a Shared Business Trajectory (SBT)[1], when an airspace user makes available the desired profile of a planned flight to the ATM System. This trajectory has previously been developed in the user organization as the Business Development Trajectory (BDT). In an iterative process between the ANSP and the airspace user and involving other affected stakeholders like airports, the SBT is adjusted based on airspace and airport capacity constraints and traffic flow requirements. In this phase, the airspace organisation is also adjusted by the ANSP (as discussed in the next section). Approaching the actual execution of the flight, a Reference Business Trajectory (RBT) is finally agreed. It is the trajectory, which the airspace user agrees to fly and the ANSP and airport agree to facilitate. Figure 1.2 shows the BT lifecycle in the CDM process.

When the RBT is executed (i.e., during the flight), all stakeholders try to meet or leverage the target 4D-spacetime points agreed in the RBT. Authorization for the RBT is given progressively during execution and takes the form of successive clearances by the ANSP (in managed airspace) or is a function of the aircraft (in unmanaged airspace). The RBT is not static during flight execution but continues to evolve based on the clearances assigned. It is continuously compared to a predicted trajectory, which is computed on-board in capable aircraft and corresponds to what the aircraft is predicted to fly. If it deviates too much for whatever reason, an automatic RBT update process is initiated with the ANSP. The ANSP or the aircraft (crew or systems) can also manually initiate an RBT revision.

[1] *Business Trajectory* is a SESAR expression referring to the "operator economically optimized flight profile" [172].

1.1.2 Airspace Organization and the 4D Trajectory

Today, flights obeying Instrument Flight Ruless in managed airspace are usually required to follow the fixed route structure of the airspace along airways and waypoints (Figure 1.3a). This airspace infrastructure, albeit entirely virtual, is a rather static construct without frequent changes, comparable to a network of roads and junctions in the sky. This is mainly due to the intricate and time-intensive distribution process of this infrastructure information through Aeronautical Information Services (AIS).

The future concept is a much more flexible one, where most of the managed airspace is essentially infrastructure-free (Figure 1.3b) and air traffic is organized based on the BTs of all aircraft sharing the airspace in a collaborative process, with airspace reservations taking place dynamically where necessary, initiated by the ANSP.

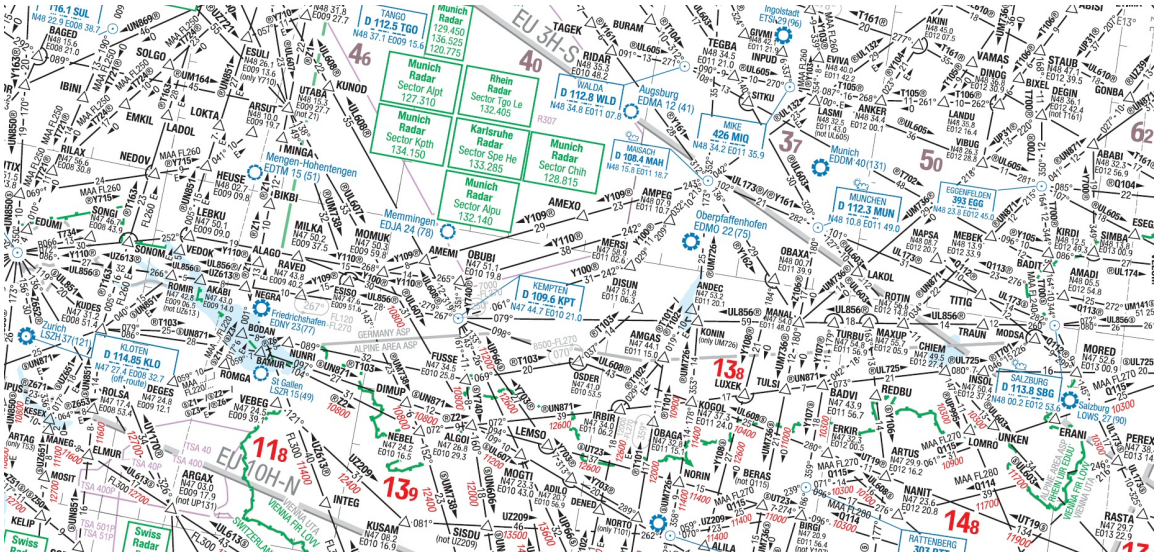
Consequently, the way a flight is registered with air traffic authorities will also change in the future. A flight plan in the currently common format of the International Civil Aviation Organization (ICAO) contains the requested departure time, cruising speed, the horizontal route as a sequence of waypoints and airways and one requested (maximum) flight level for the whole route [97]. In contrast, the BT as discussed above is envisioned to be a much more specific horizontal and vertical "4D" profile, i.e., the planned location of the aircraft in space and time, in a geometric representation [166, 172, 48]. A trajectory is thus represented by a sequence of points in space and associated timestamps, i.e., tuples (ϕ, λ, a, t), where ϕ and λ denote the geographic latitude and longitude, respectively, a the altitude, and t the time. The actual flight path of the aircraft is given by the interpolation of the points. Figure 1.4 shows as an example 40 trajectory points of a flight on August 05, 2008, that took off from Frankfurt Airport at 12:12 pm and landed at Airport Madrid-Barajas at 2:17 pm.

A functional representation of the trajectory [151] is also envisioned, but not for the closer future.

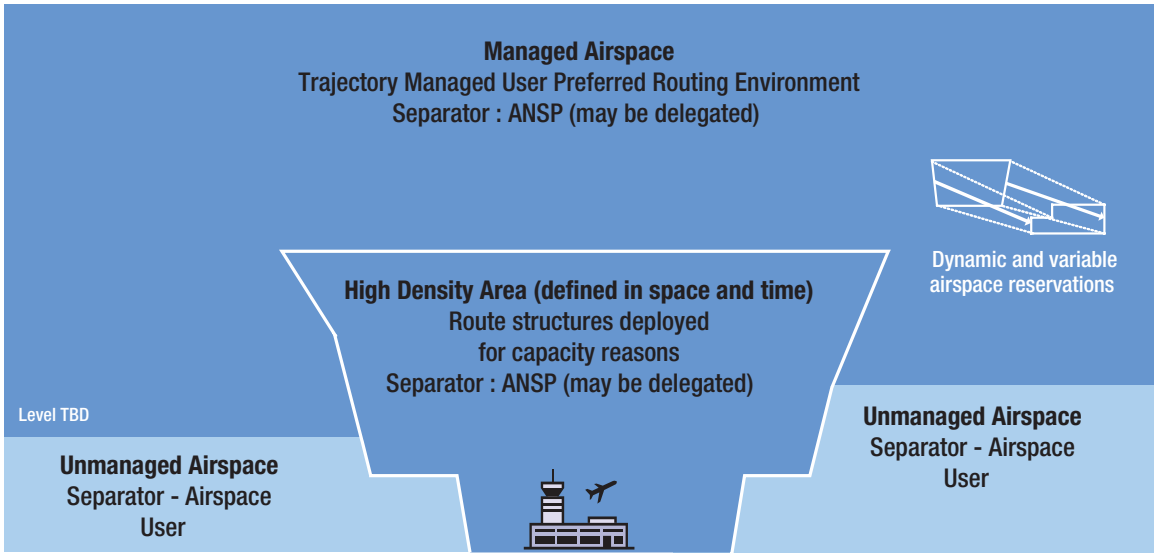
It is obvious that this approach, in combination with the CDM process, considerably raises the requirement of a timely distribution of information to all stakeholders, be it updates to the BT or dynamic infrastructure changes as well as information about all other events that could possibly affect the successful execution of the BT like weather or airspace congestion reports or forecasts. This issue is the rationale for the SWIM concept as discussed in the next Section.

1.2 System Wide Information Management

Efficient exchange of digital information plays a vital role for all of the aforementioned, fundamentally changing, processes. Timely availability of data at a much higher number

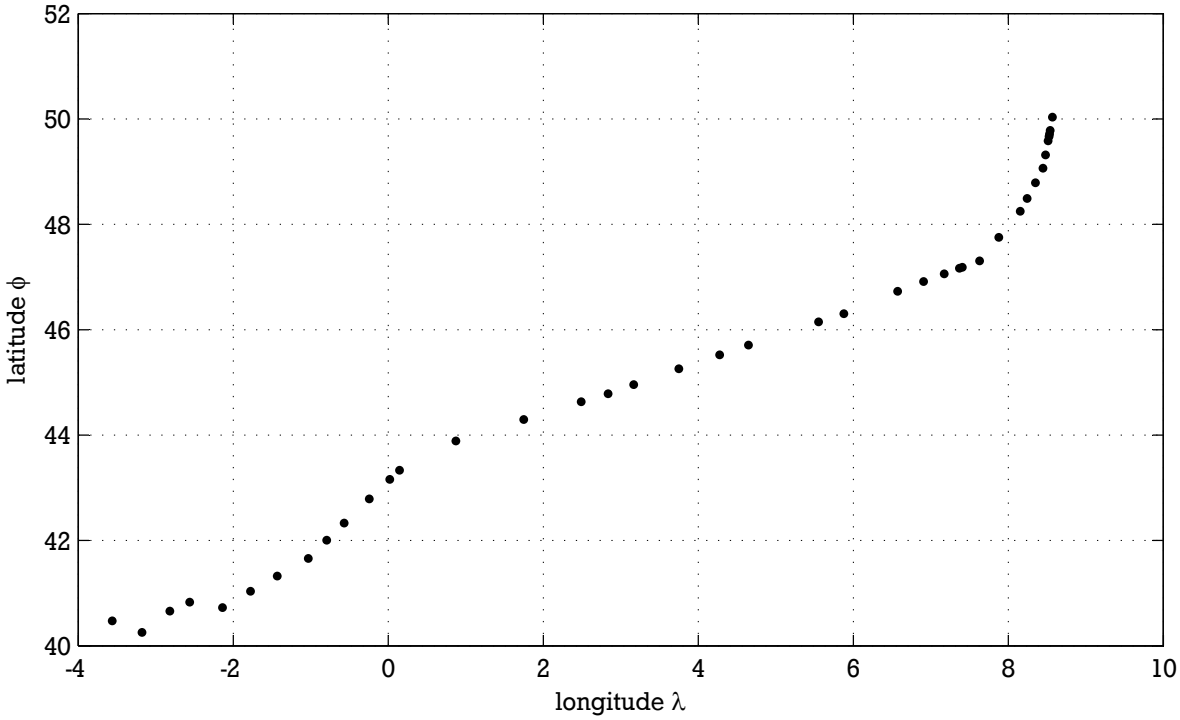


(a) Current (section of Lido™enroute chart)

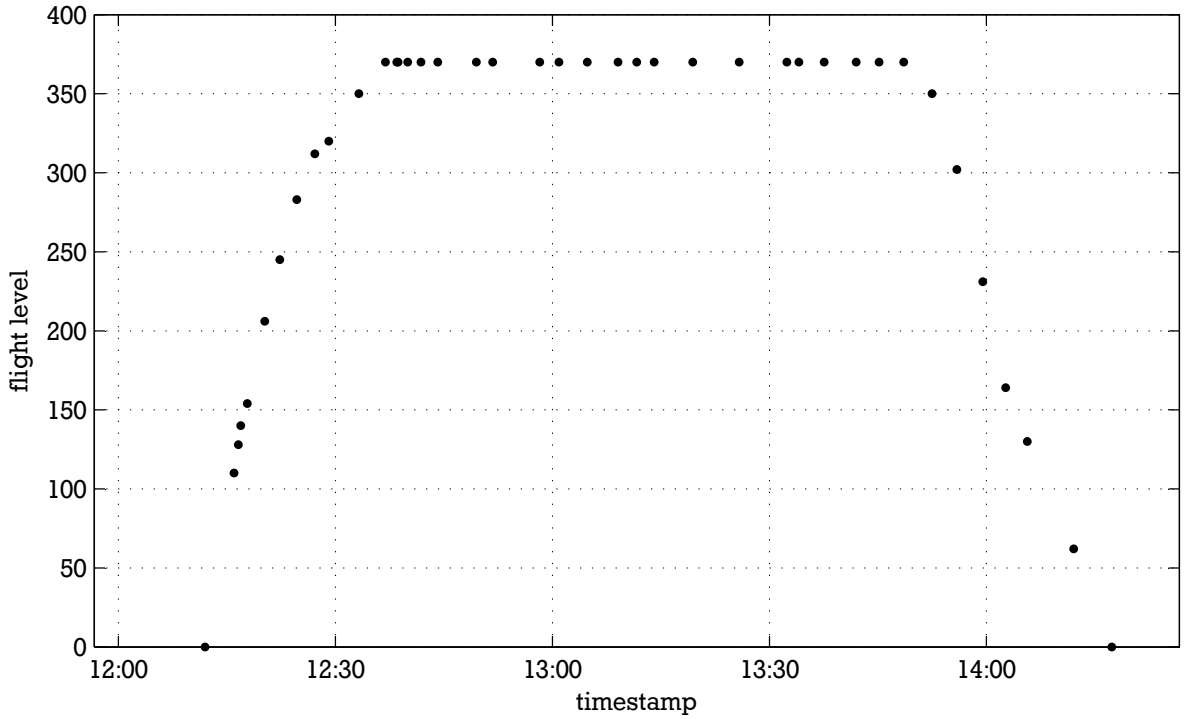


(b) Future (from [158])

Figure 1.3.: Airspace organization, current vs. envisioned future approach: static infrastructure with airways and waypoints *vs.* dynamic assignment of airspace infrastructure



(a) longitude and latitude



(b) time and altitude

Figure 1.4.: Trajectory points of a flight from Frankfurt to Madrid

of stakeholders (ANSP, ATC, airport, airline, aircraft) is a prerequisite for CDM in the planning phase as well as in the real-time environment during the execution of a flight.

Information must be easily accessible irregardless of the specific access modality or communication subsystem implementation. The heterogeneity of systems to be integrated argues for an overlay network that is built on top of legacy systems to exploit existing infrastructure while preserving legacy functionality and processes. The global distribution of the information exchange parties as well as political reasons further argue for a distributed system in favor of a centralized solution, which would limit scalability and raise single-point-of-failure concerns.

Consequently, the logical architecture in the ATM Target Concept is envisioned to be a middleware connecting geographically distributed high-level ATM systems and stakeholders' subsystems supplying ATM services, with the aircraft being a client to the system connected via air-ground data link (Figure 1.5).

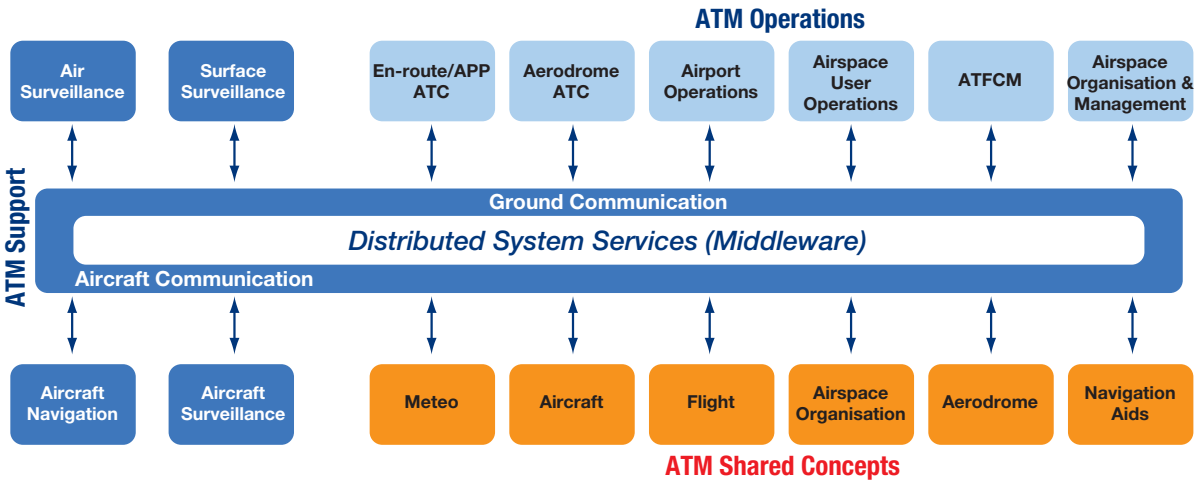


Figure 1.5.: ATM System logical architecture (from [158]).

In its core, this architecture features an information management component termed System Wide Information Management (SWIM) [164, 130, 128]. It aims to provide "value added information management services: the SWIM services. They will

- Support flexible and modular sharing of information, as opposed to closely coupled interfaces;
- Provide transparent access to ATM services likely to be geographically distributed;
- Ensure the overall consistency." [158]

Current processes of information exchange between ATM stakeholders often involve paper or voice as the means of transport for information. Where digital data exchange

systems exist, they additionally suffer from prevailing inefficient communication structures resulting from long-time proliferation, namely multiple point-to-point connections between individual stakeholders requiring custom interfaces [130, 157]. It is the pronounced goal of SWIM to get rid of these dedicated point-to-point information flows and establish "interaction between decision makers [...] through information sharing, i.e., via a distributed 'virtual' information pool" (Figure 1.6) [157].

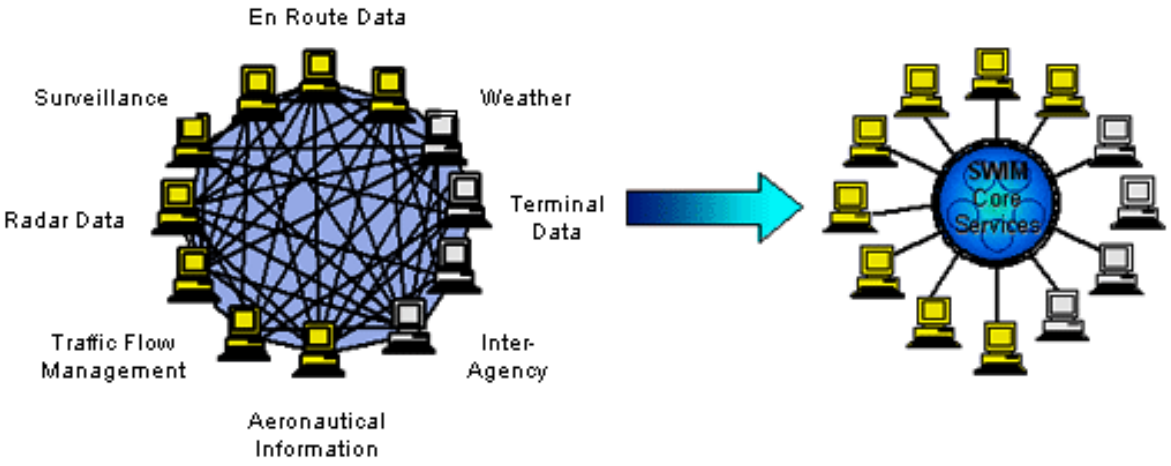


Figure 1.6.: ATM information exchange to date and in future after SWIM deployment (from [70]).

1.2.1 Event Dissemination Service

The successful execution of the BT is a key requirement in the ATM Target Concept. While the planning steps of the business trajectory (BDT and SBT) start months or even years before the actual flight, it is the time of flight conduction, when the need for timely information delivery that could affect the RBT approaches real-time requirements. Events like adjustments of the airspace infrastructure, airspace congestion reports or forecasts as well as weather reports and forecasts could make an update of the RBT necessary or even prevent its successful execution. Notifications of these events must therefore be disseminated to all affected stakeholders in a timely fashion.

Under these conditions, SWIM core services have to provide the means for a scalable event notification service that allows information to be mediated between so-called *Publishers* and *Users* of information [158]. While these services are required to be generic enough to be exploitable throughout different domains and within the heterogeneous legacy subsystems, it is envisioned to nevertheless implement them as specialized as possible in order to be easily applicable and efficient. *Publish/subscribe messaging* is declared a SWIM core capability [70, 157], implemented in a distributed systems layout

to account for the diverse ATM systems and stakeholder subsystems. Such a distributed event dissemination system (also called Distributed Event-Based System (DEBS)) with interface communications based on the publish/subscribe paradigm provides for the dissemination of event notifications between the individual stakeholders who act as clients to the system. Each stakeholder has a specific need to receive or publish event information. The event system enables them to express their interest in receiving events by subscribing to a specific type of event (acting as *User*) and to publish event notifications pertaining to a specific fact (acting as *Publisher*). The User and Publisher roles are explicitly defined in the ATM Target Concept [158], describing the stakeholders' (sub)systems, which act as the consumers and producers of information in a data exchange scenario.

Many different flavors of publish/subscribe systems have been proposed in the past and exist today as productive or prototypical implementations. The next Chapter 2 gives an overview as part of a general introduction of publish/subscribe systems. Distributed content-based notification filtering is probably the most sophisticated system layout. In this approch, Users subscribe to event notifications by means of subscription filters that are evaluated on the content of notifications. Consequently, it requires the content of notifications to be modeled precisely in order to allow for appropriate subscription filters.

In the frame of SWIM, the *ATM Information Reference Model* defines the basic principle for this content model.

1.2.2 ATM Information Reference Model

The SWIM concept requires information to be modelled explicitly, to allow a precise and concrete definition to be agreed. The SWIM services are organized around 6 data domains (Figure 1.7).

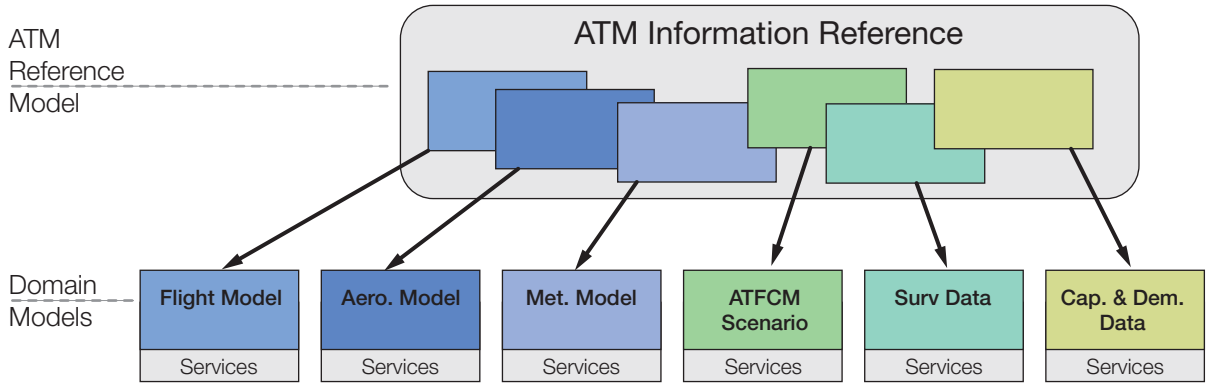


Figure 1.7.: SWIM Interoperability Models (from [158]).

The individual domains are of course in practice not completely independent, which could lead to inconsistencies among the models if they were developed regarding each domain isolatedly. This issue is addressed with an overall ATM Information Reference Model, which defines the semantics of all information to be shared and forms the master definition, subsets of which would be used in lower level models supporting interoperability for data-sharing domains [158]. The ATM Information Reference Model features domain-independent models of fundamental information concepts like spatiality (geometry, geography) and temporality (Figure 1.8).

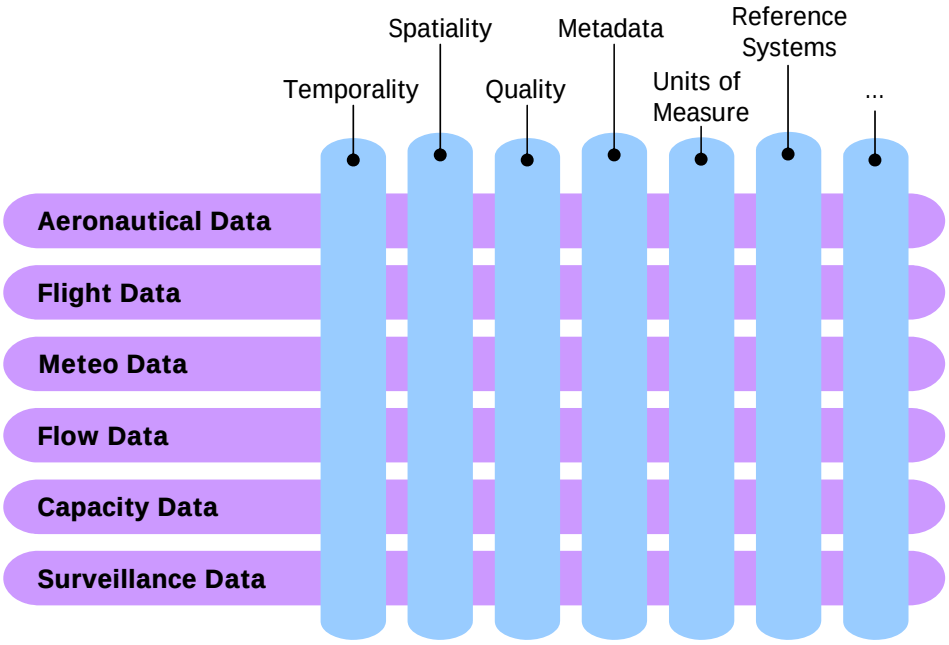


Figure 1.8.: Common information concepts for ATM information domains (adapted from [167])

Among the ATM data domains identified in SWIM, Aeronautical Information has assumed an outrider role through the development of the *Aeronautical Information eXchange Model version 5 (AIXM 5)* in the last years. Within the context of SESAR and NextGen, EUROCONTROL and FAA have signed a Memorandum of Cooperation under which a data model and information exchange format for the Aeronautical Information ATM data domain has been developed, AIXM 5. The goal was to build a modular and extensible data model that supports current and future Aeronautical Information process requirements by applying mature concepts and technologies (i.e., incorporate widely accepted standards). By this, Aeronautical Information has become the first of the data domains that a conceptual model has been developed for that meets the SWIM goal of explicit and precise information modelling. Furthermore, the fundamental concepts *temporality* and *spatiality* have been designed and integrated in AIXM 5 as

generic models as envisioned in the ATM Information Reference Model. The AIXM 5 Temporality Model comprehensively supports aeronautical processes by defining a set of versatile temporal constructs; its model for geospatial information applies standards of the International Organization for Standardization (ISO) and of the Open Geospatial Consortium (OGC).

1.3 The Goal: A SWIM Publish/Subscribe System for Aeronautical Events

On one hand, the publish/subscribe (pub/sub) paradigm's generic concept of information dissemination matches well the SWIM services' design goal of versatile core communication services. The desired specialization of the services on the other hand lies in the Users' possibilities to specify the "kinds of notifications" they are interested in. By implementing a subscription and content model based on the aviation-specific models of spatiality and temporality, a specialized pub/sub service for aviation purposes is designed. Users subscribe to the place and time they want to receive information for and Providers publish notifications for events that pertain to a specific time and space.

In this thesis, the development of such a distributed aeronautical pub/sub system is described. It builds on a distributed notification service implemented by a broker overlay network, thus capable of abstracting from the lower-level network implementation. This way, existing physical links of different kinds can be used as the data exchange connections between the brokers.

The pub/sub system builds on a content-based subscription model, in which subscriptions are implemented as filters on notification content. A spatiotemporal subscription model and an aeronautical event notification model are developed and implemented that allow for the subscription to, and publication of, aeronautical event notifications, by means of the event's spatial and temporal effectivity.

1.3.1 Contribution of this Thesis

The contribution of this thesis is threefold: an in-depth description of the temporal semantics and spatial aspects of aeronautical events, the creation of a spatiotemporal filter model for event-based systems, and the investigation and proposal of a filter merging approach for (multidimensional) interval filters in combination with appropriate filter handling algorithms for distributed event notification brokers. Beyond the implementation for the specific application of a SWIM core communication service for aeronautical events, these three aspects contribute to more generic problems as follows.

Temporal Event Semantics: While AIXM 5's Temporality Model is designed particularly with regard to AIS operations and products, it is nevertheless a generic conceptual

model of the temporal evolution of information. It is a universal model of the temporality of information in that sense, therewith providing the semantic model of time as a fundamental concept beyond the aeronautical domain.

In particular, the notion of temporally coordinated and planned events extends the common event model in the frame of event-based systems with a new aspect. In consequence, it raises a number of questions regarding the handling of temporal aspects of events, which have not been regarded previously.

Spatiotemporal Filters: In this work, spatiotemporal filters are designed and implemented as conjunctions of geometry and interval filters. This filter model is a generic one, not limited to the space and time domains. The presented formalization of filter operations and relationships as well as its implementation provide for the application to many other domains. Furthermore, while filter relationships as employed in this thesis make use of topological spatial relationship theory exclusively, other types of relationships (e.g., distance or orientation) can be used in this formalization just as well.

Imperfect Filter Merging: When large quantities of filters have to be managed, it is a consequent question whether it makes sense to somehow aggregate similar filters in order to then make the filter handling and the matching, the application of the filters on a set of data items, more efficient. The investigation of different heuristics for filter merging and the final proposal of a solution for multidimensional interval filters is a new contribution to the general question. The proposed solution leverages selectivity estimation by an assessment of per-dimension notification density and implements the concept of a quantized virtual filterspace in which every filter dimension is characterized by a normalization parameter used to homogenize different dimensions, which is a new approach in this context. Furthermore, extensions of existing filter handling algorithms for distributed event notification brokers are presented that allow for the application of the imperfect filter merging approach.

1.3.2 Organization of this Thesis

After this introduction, this thesis starts with an overview of the fundamentals of and state of the art in pub/sub systems in Chapter 2. Chapter 3 presents the models of spatial and temporal aeronautical data that support current and future information publication and distribution processes in *Aeronautical Information Management (AIM)*. It forms the basis for the content model of the SWIM pub/sub system, a spatiotemporal model of aeronautical event notifications. The subscription model of this service is based on

generic spatial and interval filters, which are introduced in Chapter 4 together with respective filter relations.

Chapter 5 exclusively treats filter merging. Heuristics are derived to assess the quality of filter mergers by developing decision functions for simple and multidimensional interval filters. The decision strategies are assessed for different combinations of subscription filter and notification sets. A “virtual filter space” approach is presented, where real filter values are mapped to discrete steps in a multidimensional virtual filterspace.

Algorithms to leverage the filter merging approach in the filter handling processes of a broker in the distributed notification service are presented and evaluated in Chapter 6. The evaluation is done based on the results of experiments carried out on realistic future flight trajectory and aeronautical event notification data.

Chapter 7 concludes this thesis with a summary of the findings and an outlook at possible future work.

2 Publish/Subscribe Systems

This chapter presents the fundamentals and state of the art of distributed pub/sub systems implementing content-based filtering and routing.

A general introduction into the basics of pub/sub is given in Section 2.1. Section 2.2 discusses a common implementation of pub/sub systems, where the functionality of the central component, the event notification service, is decentralized. The notification service's main task is the evaluation of the subscriptions on behalf of the clients. *Distributed notification routing* is the approach of sharing the computational effort of this task among the nodes of the notification service, thus aiming to improve the scalability of the system. Naïve approaches to distributed notification routing can be improved by exploiting commonalities among subscriptions, thus reducing the number of subscriptions that are exchanged over the network and have to be managed. Furthermore, the evaluation of content-based subscriptions can be quite costly, and approaches exist to aggregate and simplify subscriptions to expedite subscription evaluation at the cost of unnecessarily forwarded notifications. Section 2.3 discusses these advanced approaches to improve and optimize the tasks of notification routing and subscription management and evaluation. Section 2.4 finally summarizes the state of the art and discusses the shortcomings and missing functionality for the application of distributed pub/sub systems for the dissemination of aeronautical event notifications.

2.1 Fundamentals

The pub/sub paradigm has since its early days in the 90's gained considerable importance for push-based data dissemination, often in support of event-based systems and often in a distributed systems environment. The inherent loose coupling and the asynchronous communication style are valuable benefits in many configurations for data dissemination [29]. Among the various flavors of pub/sub systems [57], the content-based filtering approach with its inherent high expressiveness [32] has been in the focus of extensive research for several years now and many aspects of it still are, especially with respect to scaling and performance issues in large-scale pub/sub systems [152, 138, 115].

In the following subsections, pub/sub systems are first put into perspective in relation with event-based systems and contrasted with the well-known request/reply model (Subsection 2.1.1). Next, the main constituents of pub/sub systems are presented: the information to be distributed in the form of event notifications (Subsection 2.1.2),

and the components of a pub/sub system (Subsection 2.1.3): the clients, which act as producers or consumers of information, and the event notification service, which is responsible for relaying the information. Finally, the different flavors of pub/sub systems with respect to how consumers can subscribe to event notifications and how published notifications are filtered, are presented (Subsection 2.1.4).

2.1.1 Event-based Systems and Publish/Subscribe

A common architecture in networked computer systems today is the client/server layout, where the components – clients and servers – interact through request/reply communication. It is probably the best-known and most widely implemented interaction model for information exchange. In this setup, a component acts as client by requesting data or functionality and another component acts as server by providing the data or functionality.

Despite its indisputable advantages of simplicity and familiarity, the request/reply interaction model also exhibits some drawbacks. Firstly, in a data exchange scenario the designated direction of information flow (from server to client) is opposite to the direction of control flow (from client to server). The client acts as the initiator of communication requesting data hosted by the server. Considering that in most realistic scenarios of networked computer systems the information is not static but rather constantly changing, the client must continuously poll the server to stay updated, possibly resulting in unnecessary network traffic. The poll frequency merely determines the trade-off between wasted network bandwidth and update time lag. These applications would therefore benefit from a communication pattern where the server *pushes* information about state changes ("push systems" [92]) instead of requiring the client to *pull* this information. Secondly, the interaction is *synchronous*, i.e., the execution of the client's business logic is blocked while it is waiting for the server's reply. This is an intrinsic limitation of the model rather than a question of implementation. It requires application programmers to integrate business logic with interaction logic and data exchange (and communication) logic, which inherently impedes modularity and limits maintainability. In contrast, *asynchronous communication* allows the exchange of information without blocking. The component requiring data may continue processing its business logic without blocking while waiting for a reply. Thirdly, the direct communication between the components requires *explicit addressing*. The client must know where to get the data (which may be mitigated through the use of directories), must explicitly address the server in its request, and must supply its address such that the server can send the reply back. As with synchronous interaction, this requires to integrate communication

logic to a wide extent into the components. These two characteristics result in a *tight coupling* of the involved components.

Event-based systems [92, 138] take a completely different approach. Client components, which may act as *producers* or *consumers* of information, are inherently *decoupled* by a mediator, the *notification service*. Producer components *publish* event notifications, which are delivered by the notification service to those consumers who have previously *subscribed*.

		Initiator	
		Consumer	Producer
Addressee	Direct	*Request/Reply*	*Callback*
	Indirect	*Anonymous Request/Reply*	*Event-Based*

Figure 2.1.: Taxonomy of interaction models (from [72])

Publish/subscribe is the communication paradigm employed by event-based systems. The expression "pub/sub system" is often used interchangeably with "event-based system". G. Mühl *et al.* [131] and A. P. Buchmann *et al.* [27] have however noted that the model of interaction between components (event-based) is to be distinguished from the underlying communication technique (pub/sub). Event-based interaction is characterized by indirect, push-based cooperation between components (Figure 2.1, see [72] for a discussion and taxonomy of interaction models). The notification service merely provides the means to use pub/sub communication functionality, whereas the event-based interaction is mainly a characteristic of the components [138]. Figure 2.2 visualizes event-based interaction and its conceptual independence of the underlying communication technique.

The focus of this thesis is not on event-based interaction in the above sense, but rather on the requirements imposed by employing the pub/sub paradigm for the dissemination of event notifications. We will therefore in the following, as far as possible, focus on the characteristics of pub/sub systems and refrain from discussing the specifics of event-based interaction.

Pub/sub systems provide for a decoupling of producers and consumers of information in *space*, *time*, and *synchronization* [57]: Since information is exchanged in the form of notifications, which are not sent directly but mediated by the notification service, the communicating parties do not need to know each other. Neither producer nor consumer hold references to the other parties. Neither does any of the participants in an information exchange procedure know how many opposite parties contribute, i.e., how many

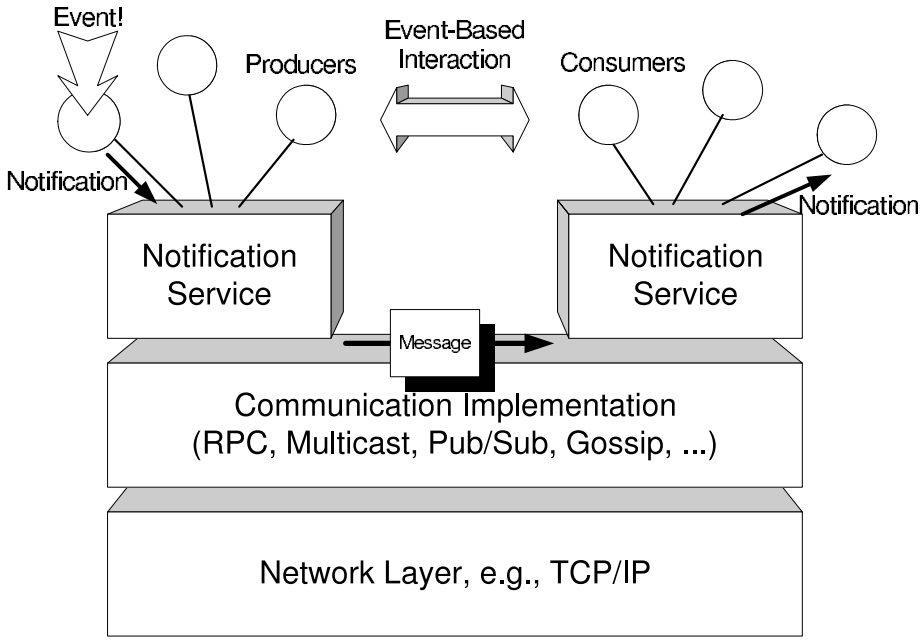


Figure 2.2.: Event-based interaction (from [180])

consumers are subscribed to, or how many producers provide, the information. This decouples the interacting parties in space. They are decoupled in time, because the parties do not need to be actively participating in the interaction at the same time. A consumer may be disconnected at the time when a producer publishes some notification and might still eventually get notified of the event[1], and the provider may publish some notifications while the consumer is disconnected. In the pub/sub system's communication layer, event notifications are encapsulated and transmitted in *messages*. *Asynchronous message passing* is achieved through message queues in each component's communication subsystem. This way, a component is not blocked during the information exchange, and the communicating parties are decoupled in synchronization.

2.1.2 Information: Events and Notifications

The information that is exchanged in an event-based system is *event* information. An event can be "any happening of interest that can be observed from within a computer".

[1] In regular pub/sub systems, the decoupling in time is limited to the network latency, i.e., the time it takes to route a notification from producer to consumer. Extending temporal decoupling requires to buffer notifications. Respective measures have been proposed by M. Cilia *et al.* [37] and A. Zeidler [180].

In the most general sense, it is a "detectable state change in a computer system" [138].[2] *Notifications* are the *reification* of events, describing the event thus that its reception and interpretation by the consumer causes a state transition in the consumer's computer system.[3] In addition, notifications model the event information thus that it can be used for filtering, i.e., consumer subscriptions can be evaluated on the content of the notification to determine if the event information affects the subscribed consumer and the notification has to be delivered to it.

The data model for notifications is called the system's *content model*. The most common content model employed in pub/sub systems is name-value pairs [31], where notifications are modeled as sets of named attributes and associated values, e.g., `{(id, "abc"), (time, 2008-03-21 12:00:00), (price, 50€)}`. Other common content models are objects [58] and semi-structured data (such as XML) [135, 14].

Throughout the remainder of this chapter, the examples used in the discussions will consistently be based on the name-value pair content model because the challenges and approaches can be explained intuitively in this model. All discussions and statements are just as well applicable to other content models nevertheless.

2.1.3 Components: Producers, Consumers, and the Notification Service

Producers are the components in a pub/sub system that publish event notifications. They are self-sufficient nodes of the system in the sense that their internal logic does not depend on the outside world. A producer is not aware of the existence of other clients. Published notifications have no adressees, they are simply passed on to the event notification service and its further processing is entirely transparent for the producers. No sort of acknowledgement message is expected from the receiver's side. It is entirely up to the notification service to distribute notifications reliably to any interested consumer.

Consumers state their interest in receiving certain notifications by providing a description of the events they want to be notified of. This description takes the form of *subscription* filters that the consumers register with the notification service. When some producer publishes an event notification that matches the subscription previously issued by a consumer, it is the responsibility of the notification service to notify the consumer of the event, i.e., to deliver the notification. Consumers, too, have no knowledge of any other communicating parties, they simply absorb notifications that are fed to them from the notification service and act according to their internal logic.

[2] The model of events required for the aeronautical event notification service described in this thesis (see Chapter 3) deviates from this definition in that aeronautical events are rather *planned* state changes in a data model of the aeronautical world.

[3] In the case of aeronautical events, notifications *may* cause a (future) state transition in the recipient's model of the aeronautical world, see Section 3.2.

Advertisements are the counterpart of subscriptions. They can (optionally, depending on the specific implementation of the pub/sub system) be issued by producers to describe the notifications they are going to publish. They can be implemented the same way as consumer subscriptions. An advertisement describes a producer's output interface. Advertisements can help improve routing decisions in the network, because the notification service knows which notifications to expect from which direction [32, 137]. Since advertisements are an optional extension not necessarily required for the functionality of the pub/sub system, we will not further discuss them in this thesis and instead refer to the respective literature, e.g., [138].

Producers and consumers are not modeled as different kinds of clients, they are merely roles clients can assume. A client connected to the system can act both as producer and consumer of event information depending on the way it interacts with the pub/sub interface. It can publish notifications or subscribe to them and get the notifications delivered it has subscribed for (Figure 2.3).

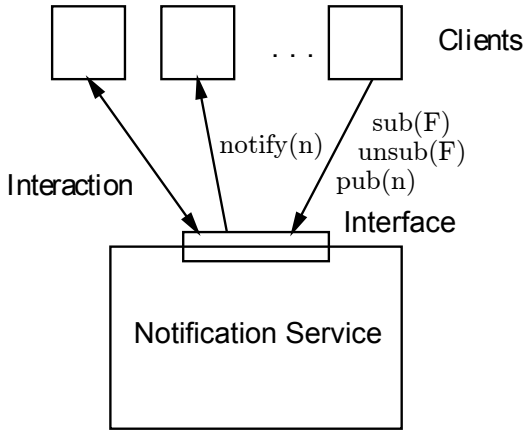


Figure 2.3.: A simple view of a publish/subscribe system (from [134])

The *(event) notification service* provides a pub/sub interface with methods implementing the required functionality:

- *pub(n)*: publish an event notification n.
- *sub(F)*: subscribe to F.
- *unsub(F)*: unsubscribe from F.

In addtion, clients register a callback function *notify(n)* when connecting to the notification service. This function is called by the notification service to notify the client of an event, i.e., to deliver the notification n.

Clients are unaware of the specific implementation of the notification service. It presents itself as a black box to the clients. As shown in Figure 2.3, the clients interact with the pub/sub interface only.

2.1.4 Functionality: Notification Filtering

To specify its interest in being notified of some sort of events, consumers register a subscription with the notification service at the pub/sub interface. The notification service stores the subscriptions, and when a notification is published, evaluates them on behalf of the consumers to determine the appropriate set of recipients of the notification. Subscriptions serve to filter notifications, and consequently, are commonly implemented as filters F. Such a subscription filter then either *matches* a given notification (or, the notification matches the filter) or it does not. A filter F can be seen as a boolean-valued function that takes as input a notification n. $F(n)$ returns either *true* or *false*, signifying whether the filter matches the notification. Let $\mathbb{N}$ denote the set of all notifications. Then the filter matching operation is defined by: $F : \mathbb{N} \rightarrow \{true, false\}$.

Various notification filtering mechanisms have been proposed in the literature and implemented in pub/sub systems. They differ in the way subscription filters are specified and the resulting expressive power: the more expressive a subscription model, the more precisely a consumer can specify its interest [17]. The following are the common subscription models:

Channels The simplest form of specifying sets of notifications is by explicitly assigning them to named channels. Producers post notifications n into a specific channel, e.g., by assigning the channel name <xyz> to it. Consumers select a specific channel by subscribing to it and will receive all notifications posted thereto. Using channels, no actual filtering is performed by the notification service. A notification is simply delivered to all clients who have subscribed to the channel. Expressiveness is very limited, because it depends directly on the number of available channels. With few channels on one hand, notification selection is very coarse. To represent the various interests of consumers on the other hand, a large number of channels have to be created.

Channels as notification selection mechanism are implemented in, e.g., the *Object Management Group (OMG) Event Service* (formerly known as the CORBA *Event Service*)[140].

Subject-Based Filtering Extending the general idea of channels, producers can label notifications with string subjects. Subjects are organised in a tree structure such that each subject string represents a path starting from the root of that tree,

e.g., `/news/sports/football`. Subscribers issue their interest by specifying the subjects they are interested in. The expressiveness of the subscription model is improved by allowing wildcards (such as `/news/sports/*`) and possibly arbitrary pattern matching. Then $F(n)$ evaluates to true if the subject subscription filter F matches the subject of the notification n. The fact that a single subject is assigned to a notification and that subjects are organized in a tree, thus disallowing multiple super-subjects, however severely constrains the expressiveness of this model.

Subject-based filtering is implemented in, e.g., TIBCO *Rendezvous*[4] [141].

Type-Based Filtering Notifications are modelled as objects of a specific type, encapsulating attributes as well as methods. Constraints are mainly expressed on the object type, thus resembling subject-based filtering. However, object types are not restricted to single super- and multiple subtypes, but – using multiple inheritance – the subject tree is expanded and allows for different paths to the same node. In addition, constraints can be specified on the attribute of the object allowing for the specification of more fine-grained constraints as in content-based filtering as described next.

Type-based pub/sub has been described by, e.g., Eugster [55, 56] and implemented in, e.g., *Hermes* [149].

Content-Based Filtering This is the most general and powerful filtering scheme available thus far. Filters specify constraints on the content of the notification, i.e., on the event information itself. It does therefore not require to explicitly assign "artificial" filtering characteristics to it like channels or subjects. Its expressiveness depends strongly on the system's content model and the means provided to express filter constraints, the *filter model*. For instance, a filter model for name-value-pair notifications could allow for the specification of arbitrary predicates on values, e.g. F ={`(price < 50€), (substr(id,1)="a"), (date=2008-03-21)`}. The general content-based filtering approach thus leverages highly expressive filter models. The more of this expressive power is enabled in a filter model by providing for the specification of constraints of diverse types, the more complex (and costly) filter handling in general and notification matching in particular can become.

Content-based filtering is employed in various flavors in several implementations of pub/sub systems, e.g., in SIENA[5] [31], JEDI [44], Elvin [155], and also the pub/sub system the extensions proposed in this thesis have been implemented

[4] `http://www.tibco.com/software/messaging/rendezvous/default.jsp`
[5] `http://www.inf.unisi.ch/carzaniga/siena/`

in, REBECA [71, 138]. A complete specification of content-based subscription models can be found in reference [133].

Concept-based pub/sub has been proposed by Cilia *et al.* [36, 15] as a general enhancement of content-based filtering. It allows to define semantic mappings between content and filter models such that, for instance, a filter $F = \{$`(price < 100$)`$\}$ can be sensibly evaluated on a notification n with `{(price, 50€)}`.

In this thesis and in particular in the following discussions, we assume a content-based filtering approach.

2.2 Distributed Systems Architecture

The clients' black box view of the notification service depicted in Figure 2.3 pertains in particular to the distribution of the architecture, i.e., whether the functionality is centralized or distributed. Obviously, centralizing all functionality of the notification service (in one "server") is easy to implement, but severely limits scalability and is prone to be a single point of failure.[6]

The alternative is a distributed notification service, in which the notification service's functionality is not implemented in a central component but by a number of interconnected system nodes as described in the next Subsection 2.2.1. To exploit the decentralization of the functionality, the main task of the notifiction service, the evaluation of the subscriptions and the provision of published notifications, is also handled in a distributed fashion. This is described in more detail in Subsection 2.2.2.

2.2.1 Notification Service

In a distributed notification service, notifications are routed through a network of interconnected brokers. This network is implemented as an overlay network built on top of some communication network (e.g., TCP/IP). Communication links between the application level brokers are established over the lower-level network layer functions.

Figure 2.4 illustrates a pub/sub system incorporating a distributed notification service based on a broker overlay network. Clients are depicted as squares and brokers as circles. The edges represent direct communication links between the components. Border brokers provide the client interface to the notification service. Inner brokers serve as

[6] In addition, a centralized implementation is not a feasible solution for a *global* aeronautical pub/sub system for political reasons. Envisioning the contribution of the air traffic authorities of a large number of diverse countries all over the world, no single place could be found that is not objectionable by some stakeholder.

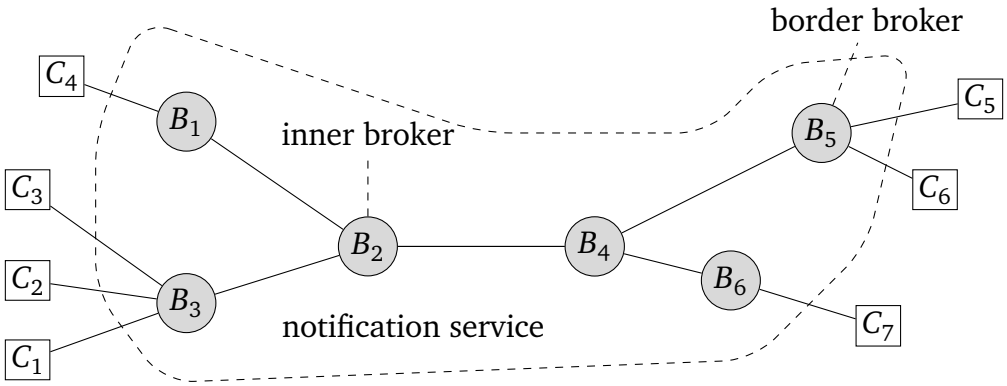


Figure 2.4.: Distributed notification service

application level routers in the overlay network. The brokers a particular broker maintains links to are called its neighbors. Every broker is aware only of the local topology of the overlay network, i.e., the links maintained to its neighbor brokers and local clients.

It is assumed in the following that the broker network exhibits an acyclic topology. This might seem like a strong limitation regarding single point of failure and scalability concerns. However, this assumption greatly simplifies some aspects of the discussion of notification routing in the next section, and alleviating solutions for performance and reliability issues have been described elsewhere [42, 149, 165]. M. A. Jaeger has further recently presented in his doctoral dissertation [106] and elsewhere [107] solutions for self-organizing broker networks that can further enhance the reliability of the distributed notification service.

The brokers communicate via messages, which are passed asynchronously via the lower-level communication links. Notifications are packaged and sent in messages, and control messages containing information about subscriptions are exchanged between the brokers. The communication links are assumed to be reliable and no messages are lost or corrupted. Furthermore, first-in-first-out ordering is respected meaning that messages are received in the same order they were sent. Albeit impractical in reality, it greatly simplifies the following discussions, and measures supporting the required characteristics can be added as extensions to the system model [42, 138].

Another common implementation of the distribution architecture that is not in the scope of this thesis but should nevertheless be mentioned because of its growing importance is peer-to-peer (P2P) overlay networks [115, 7]. In this approach the overlay network is a self-organized network of nodes forming a structured graph over a virtual key space where each key of the virtual space is mapped to a node, usually implemented as Distributed Hash Table (DHT) [165, 41]. The self-organization feature of the P2P overlay has some advantage over managed broker networks with respect to dynamic aspects of the system such as faults and churn (the continuous process of arrivals

and departures of nodes) [17]. Therefore, it is more suited in unmanaged environments characterized by high dynamicity such as mobile ad-hoc networks [179].

2.2.2 Notification Routing

The central task of the notification service is to disseminate notifications from providers to all interested consumers. In a distributed broker network this is achieved by propagation of a published notification along the broker tree. Each broker forwards the notification to a (possibly empty) subset of its neighbors.

The simplest way to guarantee that a notification reaches all interested consumers is notification *flooding*. A border broker initially receiving a new notification from a local client simply forwards it to all its neighbor brokers. Upon reception of a notification, an inner broker simply forwards it to all its neighbors except from the neighbor it came from. A border broker additionally notifies all its local clients with matching subscriptions. In this approach, subscriptions are held and evaluated by border brokers only. While this is a very simple way of notification dissemination, it obviously produces a possibly large number of notifications unnecessarily forwarded between brokers, namely in those directions where no matching subscription exists. In any case, every broker processes every notification, which severely limits scalability.

In this thesis, the commonly used *subscription forwarding* routing strategy [31, 134] is adopted instead: Each broker maintains a routing table used to decide how to forward notifications along the broker tree. A routing entry R in the table consists of a filter F and a direction D denoting the next hop, $R = (F, D)$. The destination can be a neighbor broker or a local client. Update operations on the routing table are triggered by subscriptions and unsubscriptions propagated along the broker tree. When a client C subscribes to a filter F (by calling *sub(F)* at the pub/sub interface) at the broker B, the broker adds a routing entry (F, C) to its routing table and forwards the subscription to its neighbor brokers. These brokers then each add a routing entry (F, B) to their routing table and so on. A broker's awareness of the local topology consisting of his neighbor brokers and local clients is thus supplemented with the subscription filters received from them.

Figure 2.5 shows an example. There are two routing entries in the routing table of B_1 with filters that match the notification published by C_1 (1), namely (F_1, C_2) and (F_3, B_2). Hence, the client C_2 is notified and the notification is forwarded to the neighbor broker B_2 (2).

Furthermore, client C_2 subscribes to a filter F_4 (3) and C_3 unsubscribes from filter F_2 (4). The broker B_2 adjusts its routing table accordingly (5) and forwards the subscription and unsubscription to its neighbor brokers (6).

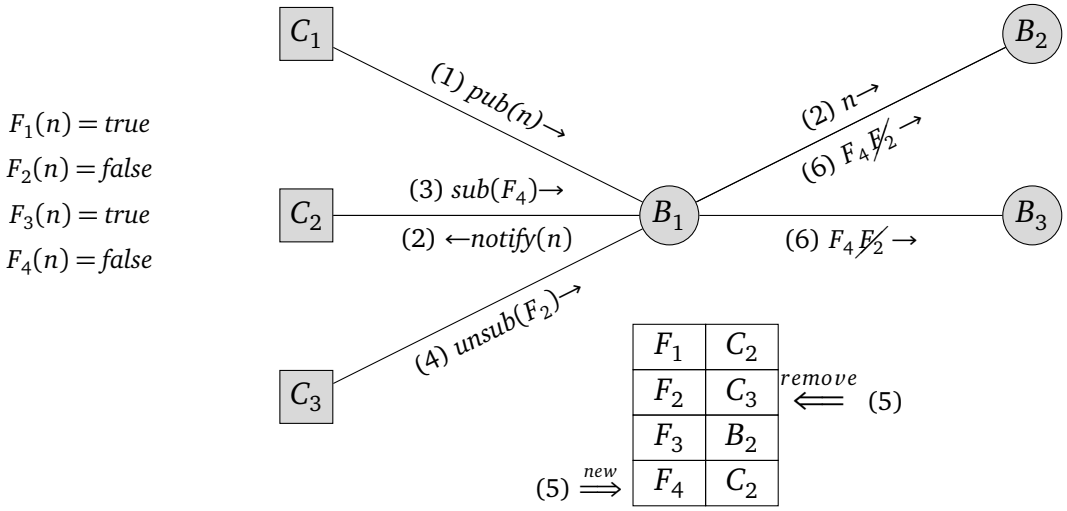


Figure 2.5.: Distributed notification routing

Notifications are forwarded in the directions where matching subscriptions exist but not in the direction from which they came. Due to the acyclic topology and the assumption of reliable communication links it is guaranteed that clients do not get duplicate notifications as long as notifications are not published twice.

2.3 Filter Handling Optimizations

Whereas in the notification flooding approach, every notification is sent to and processed by every broker, the *simple routing* approach described above can be seen as *subscription flooding*, because every subscription is processed by every broker in the network. This means in turn that every broker maintains global knowledge of active subscriptions: every broker stores every subscription. In a large-scale system with a high number of consumers, routing tables can become huge and coordination between brokers can become overly costly [136].

In large-scale systems, more advanced routing algorithms must be applied that exploit commonalities among subscriptions to reduce routing table sizes and coordination effort between brokers [137]. *Equality-based*, *covering-based*, and *merging-based* routing are such more advanced routing approaches, discussed next.

While the cost of a broker's processing of a notification, i.e., the effort required to evaluate all applicable filters on the notification, obviously depends on the routing table size on one hand, filters can be arbitrarily complex on the other hand, in form (e.g., conjunctions of constraints) as well as in the individual predicates themselves, e.g. string pattern matching or complex numerical evaluations. *Filter simplification* approaches

serve to expedite notification processing by approximating a filter with a simpler filter that can be evaluated with less effort.

The optimization approaches discussed in the following can be classified by whether or not filter quality is affected. Whereas it is a fundamental requirement for notification filtering that all notifications for which active subscriptions exist are correctly forwarded (i.e., filter evaluation must not produce false negatives), filters in a broker's routing table are in some approaches allowed to match a larger set of notifications than the original filters, thus merely conducting a candidate selection possibly resulting in falsely matched notifications (false positives), which are unnecessarily forwarded and have to be filtered using the original subscription filters later by some other broker. In the approaches presented first (Subsection 2.3.1), filter quality is not reduced, because the original subscription filters are used or new filters are created that match exactly the same sets of notifications as the original filters. Notifications are thus filtered *perfectly*. In contrast, approaches leading to *imperfect filtering* (Subsection 2.3.2) reduce subscription filter handling overhead at the cost of reduced filtering quality. Possibly falsely matched notifications result, and a trade-off between notification forwarding overhead and filter handling overhead is introduced (Subsection 2.3.3).

2.3.1 Perfect Filtering

Similarities among subscriptions can be exploited to optimize filter forwarding. The propagation and storage of subscription filters can be avoided if other subscriptions already exist that match (a superset of) the same notifications. For instance, if a broker processes a new subscription filter $F =$`{(price < 50€)}` and it has previously processed and forwarded a subscription filter $G =$`{(price < 100€)}` to its neighbors, F must not be forwarded because all notifications matching G will already be forwarded from the neighbors and this includes all notifications matching F. This way, the overall system's performance and scalability can be enhanced while still retaining the same quality of filtering [137].

The idea has first been proposed by A. Carzaniga *et al.* and implemented in SIENA [30, 31], and further researched and specified by G. Mühl *et al.* and implemented in REBECA [133, 135, 136, 134].

Equality- and Covering-based Routing

In general, two kinds of filter similarities can be exploited: *equal* filters and *covering* filters.[7] These filter similarities are described by *filter relationships*. Let $N(F)$ denote

[7] In the literature, filters matching the same sets of notifications are called "identical", and the routing optimization based on this filter property is called "identity-based routing". In this thesis, the ex-

the set of notifications matched by a filter F, $N(F) = \{n|F(n) = true\}$. Based on the relations between the sets of matched notifications $N(F)$ and $N(G)$ of two filters F and G, the following filter relationships are defined.

Definition Filter Equality: *Two filters F, G are* equal*, denoted $F \equiv G$, iff $N(F) = N(G)$.*

Definition Filter Cover: *A filter F* covers *a filter G, denoted $F \sqsupseteq G$ iff $N(F) \supseteq N(G)$. Likewise, G covers F, $F \sqsubseteq G$, iff $N(F) \subseteq N(G)$. If $N(F)$ is a proper superset of $N(G)$, $N(F) \supset N(G)$, F* properly covers *G, denoted $F \sqsupset G$.*

The routing optimization approaches based on filter similarities aim to enhance the scalability and overall performance of the pub/sub system by reducing the number of control messages exchanged between brokers and keeping the brokers' routing table sizes rather small, therewith increasing notification processing efficiency on one hand and filter forwarding overhead on the other hand.[8] However, compared to the simple routing approach without exploitation of filter similarities, checking filters for identity or covering relations introduces an addional processing step. Furthermore, the routing algorithm becomes more coplex, especially for the handling of unsubscriptions filters that cover other filters. The formerly covered filters then have to be forwarded to neighbors together with the unsubscription filter. The routing algorithm has to handle both subscriptions and unsubscriptions in an atomic step to make sure that notifications are matched correctly.

The achievable efficiency gain of these approaches thus depends the relation of the frequencies of filter handling operations (subscriptions and unsubscriptions) and notification processing on one hand, but also directly on the fraction of subscriptions exhibiting equality and covering relations. If clients tend to express diverse interests in their subscriptions and filter similarities are rare, both approaches degrade to subscription flooding, but with more complex routing algorithms. The overall performance of the pub/sub system is then likely to decrease compared to the simple routing approach. In general, the trade-off between overhead in subscription processing efficiency and reduction in the number of (un)subscriptions to store and forward has to be evaluated carefully to achieve good overall performance.

pressions "equal filters" and "equality-based routing" are used instead to emphasize the fact that two (subscription) filters $F \neq G$ have an individual identity in their own right independent of a possibly existing equality of the filter expression $F \equiv G$ with respect to the sets of notifications the filters match.

[8] The achievable reduction in routing table size is further advantageous in systems where memory utilization is an issue.

Perfect Merging

Since the achievable efficiency gain with equality and covering relations of filters depends on the fraction of subscriptions exhibiting the appropriate properties, another approach is to allow brokers to create new filters with these properties. By *merging* two or more filters $F_1, F_2, \ldots, F_n$ to form one broader, more general filter F_M, routing table sizes and the number of forwarded filters can also be reduced. F_M is created such that it covers every F_i: $F_M \sqsupseteq F_i, i = 1 \ldots n \leftrightarrow N(F_M) = \cup_i^n N(F_i)$. F_M is then called a *perfect merger* of $F_1, F_2, \ldots, F_n$ (as opposed to imperfect mergers as discussed next). With these broader filters, the chance for a covering relation with other filters also increases. The merging operation is denoted with $F_M = F_1 \sqcup F_2 \sqcup \ldots \sqcup F_n = \sqcup_i^n F_i$.

A comprehensive description of a distributed notification routing algorithm based on filter merging was first given by G. Mühl and L. Fiege [135, 136]. Specific approaches to filter merging in varying implementations of filter models have been described in, e.g., references [43, 171, 123, 163, 20].

Filter merging also introduces additional processing steps (finding merging candidates and applying the merging operation itself) for new subscriptions and the same trade-off considerations apply as with exploiting equality and covering relations. The filter handling part of routing algorithms thus becomes considerably more complex, too.

Merging of filters is not always possible (under the condition that $N(F_M) = \cup_i N(F_i)$) if the filters are not similar enough. As an example, consider the filters F_1 ={`(price < 100)`} and F_2 ={`(price ∈ [150,200])`}. Obviously, it is impossible to find a (simple[9]) filter F_M such that $N(F_M) = N(F_1) \cup N(F_2)$.

In contrast, such a merger filter can be found for F_1 ={`(price < 100)`} and F_2 ={`(price ∈ [50,150])`}, because the sets of matched notifications are overlapping. The respective filter relationship is *intersection*:

Definition Filter Intersection: *Two filters F, G are* intersecting*, denoted* $F \sqcap G$*, iff* $N(F) \cap N(G) \neq \emptyset$.

Filter *intersection* defines a "broad" filter relationship that includes other relationships, in particular equality and covering, i.e., $F \equiv G \rightarrow F \sqcap G$ and $F \sqsupset G \rightarrow F \sqcap G$ and $F \sqsubset G \rightarrow F \sqcap G$. However, two intersecting filters are not necessarily equal or covering,

[9] A possible, non-simple, perfect merger would be a disjunction of the original filters, M ={`(price = 100) ∨ (price > 200)`}. Disjunctive attribute filters are however a different topic, which is beyond the scope of this discussion. They are disallowed in content-based filter models in most cases for the complexity they add to the filter handling algorithms. This does however not limit the expressiveness of the model, because a subscriber's interest in two distinct value ranges for an attribute can still be expressed in two individual subscriptions [135, 163]. A filter model allowing for arbitrary boolean expressions has been described by S. Bittner and A. Hinze [22, 20].

$F \sqcap G \not\rightarrow F \equiv G \vee F \sqsupset G \vee F \sqsubset G$, because the enumeration of filter relationships is not exhaustive.

The opposite of filter intersection is filter disjointness:

Definition Filter Disjointness: *Two filters F, G are* disjoint*, denoted $F \not\sqcap G$, iff $N(F) \cap N(G) = \emptyset$. $F \not\sqcap G \Leftrightarrow \neg(F \sqcap G)$.*

However, disjoint filters do not generally preclude simple perfect mergers. The filters $F_1 =$`{(price < 100)}` and $F_2 =$`{(price ∈ [100,150])}` can also be merged to a filter $M =$`{(price < 150)}`, although F_1 and F_2 are disjoint. It is enough that the filter intervals are adjacent.

Finally, note that the chance to be able to merge two filters perfectly severely decreases with more filter dimensions (in the case of name-value-pair filter, that is, constraints on more than one attribute), because then it is not even enough for two filters to be intersecting in each filter dimension to allow perfect mergers [136]. We shall come back to the requirements for perfect merging in Section 5.1 with respect to the interval filters and spatial filters introduced in Chapter 4.

2.3.2 Imperfect Filtering

The optimizations presented thus far exploit filter similarities such that the filtering quality is not reduced. The required conditions to apply these optimizations are however very strict, thus limiting the scenarios in which they are applicable. Notification routing algorithms exploiting equality or covering relations among filters as well as perfect filter merging all simply degrade to subscription flooding if subscription filters do not exhibit the required relationships.

The conditions can be relaxed by refraining from the requirement that filtering be perfect, i.e., that no notifications be matched and forwarded, for which no subscription exists. In the *imperfect merging* approach, merger filters are allowed that match more notifications than the original filters. This allows to merge arbitrary filters and thus reduce the number of subscription filters to evaluate on a given notification. In contrast, *filter simplification* does not aim to reduce the number of filters but their complexity. Through approximation of subscription filters with simpler filters, the efficiency of filter evaluation is also increased.

In both approaches, imperfect merging and filter simplification, new filters are created in the process that are broader, i.e., they match more notifications, than the original subscription filters. Their application possibly results in false positives, i.e., unnecessarily forwarded notifications. Regarding the overall performance of the pub/sub system, another trade-off is thus introduced, namely between the number of unnecessarily for-

warded notifications and the number and complexity of filters to handle (forward and evaluate).

The application of imperfect filters generally requires to decide for some particular filters if it makes sense to imperfectly merge or simplify, taking the involved trade-off into account. Elaborate imperfect filter approaches therefore must include the evaluation of the quality of a created filter by considering its selectivity and the number of false positives. In any case, the merging and simplification operations themselves have to be efficiently computable in relation to the reduction in filter complexity that is achieved.

Specific approaches to assessing the quality of imperfect filters have previously been described with respect to filter mergers by A. Crespo *et al.* [43] and G. Li *et al.* [123] and with respect to filter simplification by S. Bittner and A. Hinze [24].

In the following subsections, imperfect merging and filter simplification is described in more detail.

Imperfect Merging

In the imperfect merging approach, filters can be merged such that the set of matched notifications $N(F_M)$ is a *superset* of the union of the sets of notifications matched by the original filters. F_M is an *imperfect merger* of the filters $F_1, F_2, \ldots, F_n$ if $F_M \sqsupseteq F_i, i = 1 \ldots n$ and $N(F_M) \supset \cup_{i=1}^{n} N(F_i)$.

In the example above, where perfect merging is not possible, F_1 =`{(price < 100)}` and F_2 =`{(price ∈ [150,200])}` can still be merged to an imperfect merger M =`{(price < 200)}`. This filter also matches notifications with the `price` attribute in the value range `[100,150]`, which none of the original filters matches. Hence, an imperfect merger filter F_M additionally matches a set of notifications not matched by any of F_i, namely $N(F_M) \setminus \cup_i N(F_i)$. This is the set of false positives.

Imperfect merging is theoretically always possible. However, mergers could result that are not allowed in the applied filter model (e.g., a disjunctive filter expression), or it would simply not make sense to merge the filters, because either the resulting merger would be too complex (and its evaluation more costly than the original filters) or it would potentially result in too many false positives, thus reducing filtering quality to an unacceptable extent.

Filter Simplification

All the ideas presented so far focus on routing optimziation. They potentially reduce the size of the routing tables and the number of new or canceled subscriptions to be forwarded. In contrast, filter simplification does not influence the number of stored

and forwarded subscriptions, but it also aims to expedite notification processing for the trade-off of false positives. The approach is to approximate a filter F with a filter F' that is less complex and thus can be evaluated more efficiently. F' is required to match either the same set or a superset of the set of notifications matched by F, $N(F') \supseteq N(F)$.

The availability of feasible filter simplification techniques depends on the filter model. For attribute filters in the name-value-pair model for instance, a conjunctive filter expression with constraints on many attributes can simply be truncated resulting in less constraints to evaluate [91, 21].

At the level of one broker, imperfect merging can be regarded as a special case of filter simplification. Two or more precise filters are merged into one filter of a complexity less than the added complexity of the original filters.

The imperfection introduced by simplifying F to F' is described by the set of falsely positively matched notifications in $N(F') \setminus N(F)$.

We apply filter simplification to the spatial filters introduced in the next chapter, where the spatial filter region is approximated by its bounding box.

2.3.3 Filter and Notification Forwarding Overhead

The scalability of a distributed pub/sub system that employs the simple subscription flooding approach is inherently limited, because every broker maintains global knowledge of all active subscriptions, and thus every client action (subscription, unsubscription, or publication) eventually results in a broker action for every broker of the system. Advanced filter handling schemes like equality-based routing, covering-based routing, and routing schemes applying filter merging aim to increase the scalability of the overall system through limiting the effect of client actions to subparts of the pub/sub system.

However, all these schemes can potentially degrade to subscription flooding. With the presented perfect filtering approaches, this is the case if subscription filters do not exhibit the required relationships (see Section 2.3.1). An approach based on imperfect filter merging relaxes this dependency on subscription filter relations for the trade-off of potentially unnecessarily forwarded notifications, thus theoretically running the risk to degrade to notification flooding (when all subscriptions are merged resulting in a merger covering the whole filter space). Hence, while potentially reducing *filter forwarding overhead*, the achieved benefit must be carefully balanced with the reduction of filtering quality leading to *notification forwarding overhead*.

2.4 Requirements for a SWIM Publish/Subscribe System

The support for data exchange between Providers and Users based on the pub/sub paradigm implemented as a SWIM core service has been identified as a key requirement

in SESAR as well as NextGen. Publishers assume the role of producers, and Users the role of consumers in the pub/sub system. The implementation of SWIM services is supposed to integrate as far as possible the globally distributed legacy systems of the stakeholders. These legacy systems as well as the interconnecting networks and direct communication links are highly heterogeneous. A pub/sub system with a distributed notification service implemented as a broker overlay network exhibits a readily suitable system design meeting the requirements. Brokers and access points of the pub/sub system are implemented within or on top of legacy systems, and the brokers' communication network is overlayed on, and thus abstracts of, the heterogeneous legacy lower-level communication links.

While the state of the art in pub/sub systems readily provides for such an implementation that meets the system level requirements, it lacks support of aviation-specific aspects of a pub/sub system as a SWIM core service: a content model for aeronautical events, and a filter model capable to subscribe for the events' spatial and temporal effectivity, to enable Users to subscribe for the Business Trajectory of a (planned or ongoing) flight. In such an application scenario, filter handling optimizations based on perfect filtering fail, because no two subscriptions ever cover each other (because this would mean that two flights take place in the same space and time) and are barely ever equal (only when two different Users subscribe to the same flight). Hence, imperfect filtering approaches must be applied to optimize filter handling. These open issues are detailed in the following.

2.4.1 Spatial and Temporal Filter Model

The task of a SWIM pub/sub system is to mediate notifications for ATM events from SWIM Publisher to User systems, i.e., producers and consumers in the frame of pub/sub systems. Such a system is required to provide the means that allow a User to subscribe to notifications for all events that could affect a flight. Consequently, this requires the pub/sub system's filter model to allow for subscriptions to a flight's 4D trajectory. Moreover, User subscriptions for ATM events could potentially pertain to generic sections of airspace and time (e.g., for surveillance reasons). Hence, a general requirement for a SWIM pub/sub system is that it provides a subscription model with spatial and temporal (or spatiotemporal) filters.

In Chapter 4, we introduce *(geo-)spatial* and *interval* filters as generic filter types, which provide the means to implement filters for a 2-dimensional spatial region, an altitude interval, and a temporal interval. The spatiotemporal filters constructed as a conjunction of the basic filters allow to subscribe to a 4D trajectory or any other section of space and time.

2.4.2 Aeronautical Event Model

The SWIM pub/sub system requires a content model suitable for and specific to ATM events, which should be as generic as possible while taking into account the specific requirements of aviation-related information in ATM data. In particular, the content model is required to model notifications such that they expose an ATM event's spatial and temporal effectivity, i.e., the space and time an event affects, to allow for the evaluation of respective subscriptions and the matching of spatiotemporal filters.

While the ATM data domains defined in SWIM focus on different ATM aspects, the Aeronautical Information domain assumes a unique position because its subject is the aviation infrastructure itself, i.e., the constituents of the aviation world like airways, airspaces, airports, in which all other ATM-related activities takes place. Approaching a model of spatiality and temporality of ATM events, we therefore focus on the Aeronautical Information domain, and concentrate on aeronautical events.

The next Chapter 3 derives a model for aeronautical event notifications by examining the Aeronautical Information domain with respect to its subject, the aviation infrastructure, implemented in a data model for aeronautical features integrating a generic, yet aviation-specific, spatial model, and its processes, a detailed analysis of which shows that aeronautical events, being future events at the time of notification publication, exhibit different temporal semantics than the observed events traditionally assumed in the frame of event-based systems. This requires to explicitly model the temporal effectivity of events in notifications. In particular, the analysis shows that aeronautical events can have a duration, being effective not at an instant but throughout a temporal interval.

In general, aeronautical event notification cannot be modeled as points in the filter space but can have an extent, in the spatial dimensions as well as in the temporal dimension. This is different from the common assumption in event-based systems, where events (and notifications) are conceived as (dimension-less) points in the filter space (see, e.g., reference [170]).

2.4.3 Filter Aggregation

Filter handling optimizations that rely on perfect filtering require subscriptions to exhibit covering or equality relations to be effective. Regarding spatiotemporal subscription filters pertaining to flights, these relations hardly ever exist, because flight trajectories (hopefully) never assume the same or overlapping sections of space-time. This leaves only imperfect filtering approaches as applicable optimizations in this application scenario.

In addition, political and security reasons speak for a masking of flight trajectory filters. Many stakeholders will not want their precise flight planning data to be publicly available. Since ATM stakeholders such as airlines, ANSPs and airports are often operated by or closely related to state and government institutions on one hand as well as competing with respective organizations in other countries on the other hand, the potential availability of precise flight planning data at (politically or economically) competing organziations would not be acceptable to many stakeholders. Some "non-cooperative countries" are known in the global AIS community for being very restrictive with the publication of aeronautical information, and for deliberately limiting the quality of the published data. ATM authorities of these countries would not contribute to or take part in a global ATM data dissemination system that potentially unveils precise flight planning information. This concern can be alleviated by providing for imprecise filtering in the SWIM pub/sub system implementation. It is then open to the organization or state governing a subpart of the overlay network to publish broad or aggregated, and thus masked, filters to the rest of the network.

While notification routing algorithms based on imperfect filtering have been researched and described before, it remains an open research topic —not limited to our specific application scenario of a SWIM pub/sub system— to evaluate how filter aggregation and simplification can be applied sensibly.

After introducing filter merging for spatial and interval filters, we investigate filter merging approaches in general and propose and evaluate specific solutions in Chapter 5. We apply filter simplification for spatial filters, and imperfectly merge multidimensional interval filters. Different merging strategies are formally developed and investigated as well as evaluated and compared with respect to experimental results.

3 Space and Time of Aeronautical Events

Aeronautical events are happenings in the domain of *Aeronautical Information*. Aeronautical Information is an ATM information domain defined in *Annex 15 to the Convention on International Civil Aviation* of the ICAO [98]. This Annex obligates member states to maintain *Aeronautical Information Services* (AIS) "to ensure the flow of information/data necessary for the safety, regularity and efficiency of international air navigation."

Aeronautical Information describes the aviation environment through *aeronautical features* and their *characteristics*, for instance navigational aids (navaids), airports, airways, airspaces, obstacles[1], and terminal procedures (arrival, approach, and departure) for airports. The characteristics of, e.g., a navaid are: identification, location, type, frequency, etc. Simply speaking, aeronautical events are happenings affecting aeronautical features: The creation or withdrawal of features or changes of a feature's characteristics.

This chapter describes the model of aeronautical events for the purpose of dissemination in the aeronautical pub/sub system, where event notifications expose the event's spatial and temporal effectivity for which Users can subscribe and based on which they are routed through the broker network.

Section 3.1 introduces AIS and gives an overview of AIS products and processes. Aeronautical Information processing and distribution procedures, which require to notify all Users of these happenings in advance, impose requirements on a model of aeronautical events with respect to its temporal aspects. These issues are discussed in Section 3.2, where a comprehensive model of Aeronautical Information temporality is presented including aeronautical events as core elements.

The spatial aspect of Aeronautical Information is evident: some aeronautical objects like airspaces and routes are purely virtual features defined in spatial terms, but also non-virtual objects like obstacles are mostly relevant for their spatial aspects, location and height. The spatial model for Aeronautical Information and its design based on widely accepted international standards for geospatial information are discussed in Section 3.3.

[1] Obstacles in the aviation-specific sense are defined as "All fixed (whether temporary or permanent) and mobile objects, or parts thereof, that are located on an area intended for the surface movement of aircraft or that extend above a defined surface intended to protect aircraft in flight." [98]

3.1 Background: Aeronautical Information Services

Each ICAO member state is required by Annex 15 to publish Aeronautical Information in a handbook called *Aeronautical Information Publication (AIP)*. Publication and distribution of amendments to the AIP (AMDTs) is regulated in a concept termed *Aeronautical Information Regulation And Control (AIRAC)*. AMDTs have to be distributed 42 days (56 in the case of a major change) before the information contained (i.e., the change) becomes effective. This ensures the timely distribution of information about planned changes to all stakeholders and the integration of these changes in data products like pilot handbooks, aeronautical charts, and onboard databases. Furthermore, a fixed 28-days cycle of *effective dates* defines those dates at which a change can become effective. This ensures that all users work with the same data at the same time. The recurring publication and effective dates are referred to as the *AIRAC Cycle* (Figure 3.1).

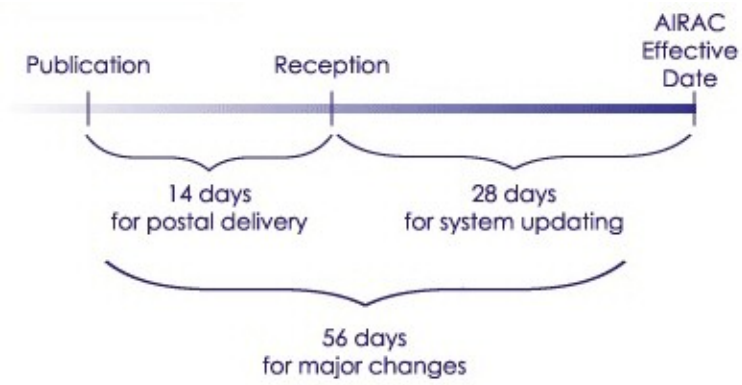


Figure 3.1.: AIRAC Cycle as defined in Annex 15 (from [147]).

3.1.1 NOTAM

There are two types of situations that cannot be dealt with in the AIRAC cycle system: (1) short-term changes that cannot wait until the next effective date and (2) temporary changes, which do not last over at least one AIRAC cycle.

A typical situation would be a damage in the runway surface that is observed that requires the runway to be closed for the reparation work for one week. The closure cannot wait until the next feasible effective date (which may be 55 days in the future), and it would not even make sense to incorporate the closed state of the runway in an AMDT, where it is effective for at least 28 days, if the maintenance work takes only one week.

So whenever an operationally significant change to the aeronautical environment must be announced that does not fit in the AIRAC cycle, a *Notice to Airmen (*NOTAM*)* is

published. NOTAM are short telegram-style text messages, intended for aviation personnel (Figure 3.2). In the case of a short-term notice of a lasting new state (a *permanent change*), the information is incorporated in the AIP with the next AMDT and the NOTAM is canceled, while in the case of the announcement of a state that is only temporary (a *temporary change*), the information is never incorporated in the AIP.

```
(A2870/05 NOTAMN
Q)EDFF/QMXLC/IV/M/A/000/999/5002N00834E005
A)EDDF B)0812192130 C)0812230500 EST
E)TWY N BTN TWY M AND TWY Q CLSD.)
```

Figure 3.2.: A NOTAM informing of a taxiway closure at Frankfurt airport

The Q-line (qualifier line) contains information intended for a coarse filtering of NOTAM, most notably the lowest and highest flight levels affected (000 and 999 in the example, meaning all flight levels), and the affected area in the form of a point (here: 50°02nnorthern latitude, 8°34ëastern longitude) and a radius (here: 5 nautical miles). In today's operations, this meta-information is usually assigned manually and is very coarse and often wrong. The indicated circle is often too coarse an approximation of the affected area and a radius values of 999 is often indicated, e.g. when the NOTAM affects the originating country's whole airspace (see Figure 3.3) [63, 67].

The begin and end of the effectivity interval of the announced state are indicated in the NOTAM's `B)` and `C)` field, respectively. The end of effectivity may be marked as "estimated" (`EST`), which means that another NOTAM will be issued later concerning the same fact, which then either cancels the previous NOTAM and thus abrogates the formerly announced situation or contains an exact end date. If a permanent change is announced in the NOTAM, the `C)` field contains the keyword `PERM`.

3.1.2 Static and Dynamic AIS Data

The state that is represented in a database containing AIRAC cycle data (or: AIP data) has traditionally been called the *static* state, and the AIRAC cycle data, *AIS Static Data*, whereas NOTAM information has been referred to as *AIS Dynamic Data* [97, 63, 67].

It is obvious that NOTAM information has precedence over AIP information, because NOTAM are supposed to notify of temporary changes, i.e., deviations from the baseline information in the AIP. The fact that the information stated in a NOTAM overrules the one stated in the AIP is common practice. Its acknowledgment by all aviation stakeholders assures that everybody is "on the same page". It is thus a crucial requirement for air traffic safety.

Figure 3.3.: Affected area circle of a Ukrainian NOTAM centered in Odessa.

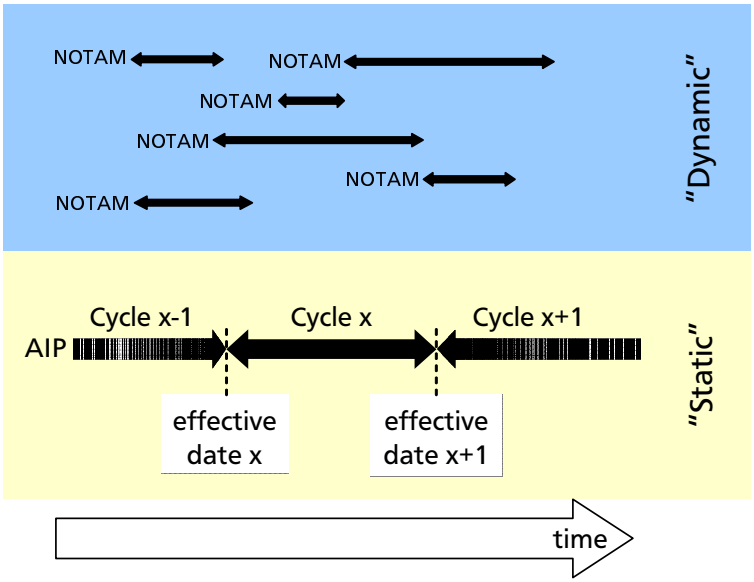


Figure 3.4.: AIS static and dynamic layers

Static and dynamic information, AIS and NOTAM, are perceived to be on different, prioritized layers. Information on the dynamic layer overrules information on the static layer. For the effectivity time of a NOTAM concerning a specific fact, the NOTAM information overlays the AIP information. After the end of the effectivity period of the NOTAM, the AIP baseline information is valid again. Figure 3.4 illustrates this concept with the AIS products AIP and NOTAM on prioritized information layers.

3.1.3 From AIS to AIM

In recent years, there has been a growing demand for an integration of AIS Static and Dynamic Data and eventually an abandonment of the fix AIRAC cycle [86], which is altogether seen as a shift from a *product-centric* view (AIP/AMDT and NOTAM) towards a *data-centric* view (aeronautical features and operations); a paradigm shift from Aeronautical Information Services to *Aeronautical Information Management (AIM)* [61, 62].

This shift involves a redesign of current AIS operations on one hand, but its key enabler on the other hand is the latest version 5 of the Aeronautical Information eXchange Model (AIXM), and its intrinsic integrated view of Aeronautical Information irregardless of the product.

AIXM is an open, publicly available conceptual model (implemented as UML class model, "AIXM CM") of and an XML exchange schema ("AIXM Schema") for Aeronautical Information. It has been developed by European Organisation for the Safety of Air Navigation (EUROCONTROL) since 1997. The European AIS Database (EAD)[2] is based on AIXM CM, currently undergoing the migration from AIXM 4.0 to 4.5.

AIXM until version 4.5 focuses on AIS Static Data (AIP/AMDT) only. AIS Dynamic Data (NOTAM) are not supported. The main reason for this is their text-based nature. A NOTAM's main information is contained in item E) in unstructured cleartext only, which makes it impossible for a computer to automatically unambiguously interpret the information and link it to the features published in the AIP. NOTAM have to be read and interpreted by a human to get the full picture, i.e., the actually currently valid state of a feature. The integrated handling of static and dynamic data was thus a key requirement for the latest version 5 of AIXM.

AIXM 5 has been developed since 2005 under a Memorandum of Cooperation between the FAA and EUROCONTROL, with the support of the international AIS community.[3] Enabling "computer-interpretable NOTAM" in AIXM format (called "Digital NOTAM" or "xNOTAM" [62, 85]) required on one hand to extend the CM such that NOTAM content can be modeled. This has turned out to be a fairly challenging task, which has been

[2] http://www.ead.eurocontrol.int/eadcms/eadsite/index.php.html

[3] This version is also proposed as ICAO standard [99].

coped with for some individual domain parts in different studies, e.g. for aerodrome data in [85] and for terminal procedure data in [79]. On the other hand, an integrated model of the temporal evolution of Aeronautical Information was needed, which still supports the traditional AIS products [68, 83]. This temporal model is presented in the next Section 3.2.

In addition, the spatial aspects of Aeronautical Information such as the location of a navigational aid or the boundary of an airspace were modeled in AIXM until version 4.5 as attributes of the respective objects in a proprietary way, preventing the interpretation, manipulation, and visualization of these data with standard software for spatial data, such as Computer Aided Design (CAD) or Geographic Information System (GIS) tools. This was to be overcome with AIXM 5 by modelling the spatial aspects based on widely accepted international standards for geographic information [9]. Consequently, the AIXM 5 CM is based on the ISO 191xx series of standards for geographic information (most notably, ISO standard 19107 "Geographic Information – Spatial Schema" [102]), and the AIXM 5 Schema (an XML Schema implementing the CM) is based on the Geography Markup Language (GML) [144, 118], which has recently become ISO standard 19136 [104].

Following these standards for GIS interoperability, the aeronautical world is abstracted in the AIXM 5 CM by means of *features* and *feature attributes*. Geographic features are spatially-referenced objects: objects in the modeled world having spatial properties like position or extent as well as possibly non-spatial properties like color or name.[4] The properties are represented in *feature attributes*.

The specifics of the spatial model are detailed in Section 3.3.

3.2 Temporal Model

A temporal model of AIS is required to enable the communication of past, present, and future facts, like the creation of a new airspace planned for the future, the temporary closure of a runway, or the erection of an obstacle for a specific time. In particular, it requires to take into account the requirements of AIS processes and support the traditional AIS products, AIP and NOTAM, as well as AIRAC operations, whereas at the same time allowing for an integration of the information without conceptual differences.

In this section, the temporal requirements of aeronautical data dissemination processes are investigated. Subsection 3.2.1 first introduces basic representations of time: temporal instants and intervals and their implementation in continuous and discrete time models. Subsection 3.2.2 then starts with the development of a conceptual model of the temporal evolution of the aeronautical world by means of feature *states* and

[4] Features are defined as "abstractions of real-world phenomena" [100].

events. In Subsection 3.2.3, temporal semantics of aeronautical events are discussed and an event model is derived from the requirements identified in an investigation of the AIS data distribution processes featuring two types of events: instantaneous events (taking place at one moment in time) and temporary events (being valid over a time interval). Being a new contribution to the research field of event-based and publish/subscribe systems, this temporal model of (aeronautical) events is presented in depth with its derivation and put in relation with other work described elsewhere.

3.2.1 Representations of Time

The expressions for and use of temporal concepts in the following complies as far as possible with the definitions given in the *Glossary of Temporal Database Concepts* by Jensen et al. [109] or, where differing, its later version [49].

Time is perceived on a directed, endless, continuous axis ranging from the past to the future, the *time axis* or *time-line*. What makes this notion specifically unique is the concept of the current time, which we term *now*. It separates past from future and is constantly moving. This specific construct has been termed "the moving point now" [46].

Now can be seen as a moving tick on the time-line. Such a tick on the time axis is called an *instant*. An instant is usually described by a *timestamp*, which maps the time-line to some *calendar*, e.g., $t_0 =$`'2008-03-21 00:00:00'` with respect to the Gregorian Calendar.[5] Timestamps and instants are strictly ordered with respect to the time-line. Given two valid timestamps t_0, t_1 in some calendar, one of $t_0 < t_1$, $t_0 = t_1$, $t_0 > t_1$ holds.

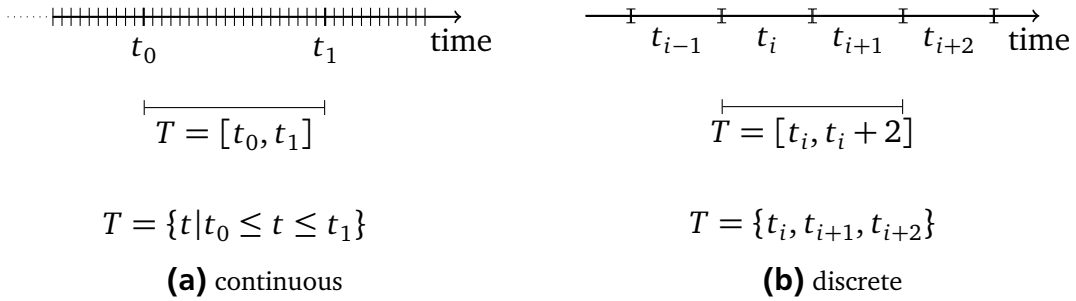
While time can be regarded at different levels of granularity (e.g., days, hours, minutes, seconds etc.), an instant is 0-dimensional in the sense that it marks the point on the time-line between those elementary calendar units. The above timestamp for instance denotes the exact instant when a second-precision clock switches from `2008-03-20 23:59:59` to `2008-03-21 00:00:00`. It is the same instant when a minute-precision clock switches from `2008-03-20 23:59` to `2008-03-21 00:00`.

Instants and timestamps provide the possibility to denote time periods or *temporal intervals*. These intervals mark a range on the time-line. They can be denoted with the instants limiting the range, e.g., the temporal interval $T = [t_0, t_1]$ with t_0 as before and $t_1 =$`'2008-03-21 00:00:01'` denotes the first second of March 21, 2008.

For the implementation of time, two different models can be distinguished: continuous and discrete models of time. In a continuous time model, instants can be specified

[5] Different calendars and time zones, leap day/year as well as ways to measure time are big topics in their own right, which are however beyond the scope here.

at any time, i.e., timestamps can be defined in an arbitrary precision. For any two instants $t_0 < t_1$, there is an infinite number of instants t_i such that $t_0 < t_i < t_1$. Hence, the time-line is isomorph to $\mathbb{R}$. Consequentially, temporal intervals can be regarded as infinite sets of instants. In a discrete time model, time is not perceived as a continuum but is regarded as a sequence of succeeding, non-decomposable time intervals of some fixed, minimal duration, so-called *chronons* [160]. Chronons divide the time-line with a specific resolution. A chronon can for instance be a milisecond, second, minute, day, or even century, depending on the requirements of the application.



(a) continuous **(b)** discrete

Figure 3.5.: Models of Time. In the continuous model, an interval contains infinitely many instants. In the discrete model, an interval contains finitely many time points.

In such a model, the time-line is regarded as an infinite sequence of *time points* and is isomorph to $\mathbb{N}$. As a consequence an instant is represented by a specific time point and timestamps denote such time points. Assuming one second as chronon, the timestamp `2008-03-20 23:59:59` would in that case for instance represent the last second of March 20, and `2008-03-21 00:00:00` the first second of March 21. It also holds for time points t_0, t_1 that they are well-ordered with respect to the time-line, i.e., if $t_0 \neq t_1$ then one of $t_0 < t_1$ or $t_0 > t_1$ is true and vice versa. Furthermore, it follows that for every time point t_0, there is a unique predecessor $t_0 - 1$ and a unique successor $t_0 + 1$ on the time-line.

The notion of an instant changes such that it now actually has an extent: the duration of a chronon. The notion of temporal intervals also changes such that they become finite sets of time points. Hence, the interval $T = [t_0, t_1]$ is the set of all time points $\{t_0, t_0 + 1, t_0 + 1 + 1 \ldots, t_1\}$ (Figure 3.5).

With that notion, the use of right-open and/or left-open intervals, $T = (t_0, t_1], T = [t_0, t_1), T = (t_0, t_1)$, has some advantegeous aspects. Then the set of time points in T does not include the respective endpoints. Two reasons argue for open intervals: Firstly, the meaning of $T = [t_0, t_1)$ is more intuitive, when t_0, t_1 are specific timestamps. For instance, the interval $[$`2008-03-20 17:00:00`, `2009-03-22 00:00:00`$]$ includes the first second of March 22, which is counter-intuitive. We would assume that

the interval ends at midnight. This would exactly be the case with the right-open interval $\left[\texttt{2008-03-20 17:00:00}, \texttt{2009-03-22 00:00:00}\right)$, where the latest second included is the last second before midnight. Secondly, dividing a right-open interval $T = \left[t_0, t_1\right)$ into two disjunct intervals $T_{\text{lower}}, T_{\text{upper}}$ at some time point $t_\text{d}, t_0 < t_\text{d} < t_1$ it can be intuitively defined that the resulting intervals are $T_{\text{lower}} = \left[t_0, t_\text{d}\right)$ and $T_{\text{upper}} = \left[t_\text{d}, t_1\right)$, i.e., that t_d belongs to the upper interval. The opposite would be an intuitive definition in the case of left-open intervals. In the case of the closed interval $T = \left[t_0, t_1\right]$ it is not intuitively clear to which of the two resulting intervals t_d belongs.

In the following development of a conceptual temporal model, closed intervals will be used continuously for ease of notation and readability. Nevertheless, they could be changed to left-open or right-open intervals without effect on the discussion. We will resume using right-open intervals for the presentation of the actual implementation of the aeronautical event notification model in Section 3.4.

3.2.2 Aeronautical Feature States and Events

Aeronautical features are created at one point, undergo various changes over their lifetime and cease to exist at some point in time. When features change, it means that some modeled characteristic of the real-world object changes, which in turn means that the value of some feature attribute changes.

At any specific moment in time there is however exactly one set of feature instances, i.e., one set of existing features with defined attribute values. Such a set of feature instances describes a *state* of the modeled world. When something happens in the real world that results in a change in the modeled world, a state transition takes place: The validity of one state ends and the validity of another one begins. A state transition is understood as happening instantaneously, i.e., at an instant. The happenings in the real world that cause state transitions in the model are called *events*.[6]

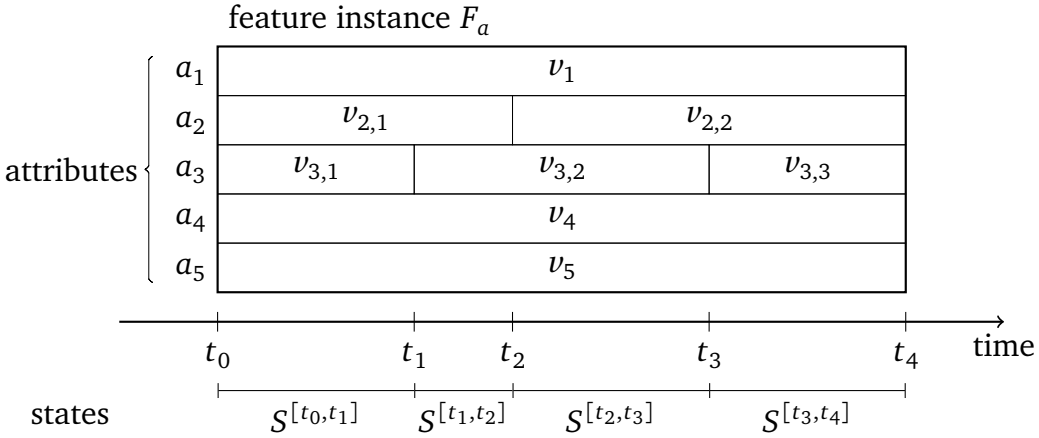
States are valid throughout a temporal interval, the begin and end of which are event times. The event time t_e of the event E^{t_e} marks the end of the valid time interval of one state $S^{[\ldots,t_\text{e}]}$ and the begin of the valid time interval of the next state $S^{[t_\text{e},\ldots]}$.

There are three conceptually different kinds of events:

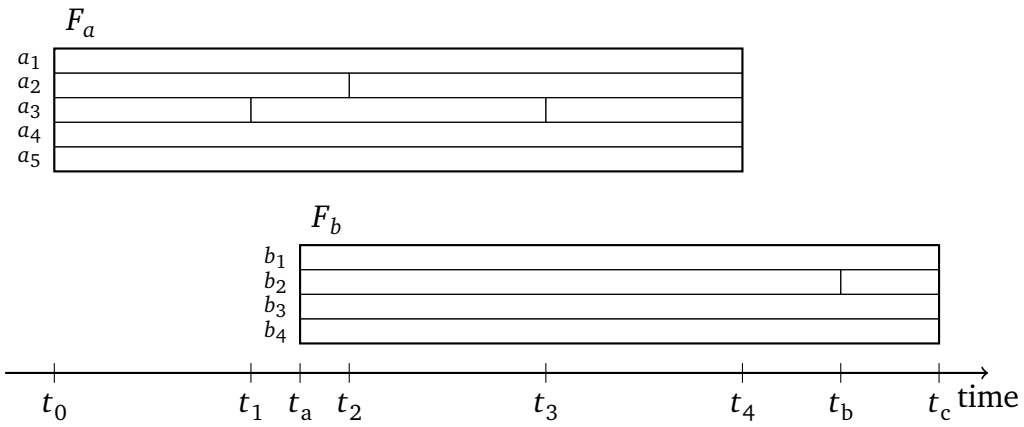
- Creation of a new feature (e.g. assignment of a new airspace)
- Change in a feature's properties (e.g. change of the frequency of a navigational aid)
- Withdrawal of a feature (e.g. abandonment of an airway due to restructuring)

[6] State transitions and events are often not differentiated. ISO standard 19108 [101] for instance speaks about events only and defines an event as "action which occurs at an instant".

Figure 3.6a illustrates states and events regarding one feature over its lifetime. The feature instance is created at t_0. It has five attributes, $a_1, \ldots, a_5$, the values of which change over the feature's lifetime (at event instants t_1, t_2, t_3). At t_4, the feature is withdrawn. Figure 3.6b illustrates states and events regarding the modeled world.



(a) One feature's lifetime



(b) The modeled world's evolution

Figure 3.6.: Temporal evolution of the modeled world. Events occur at the instants marked on the time axis. A state is valid between events.

Since the begin and end times of a state's validity interval mark events and in turn, states are valid from one event to the next, there cannot be an event within the validity interval of a state.

3.2.3 Temporal Semantics of Events

Having prepared the conceptual foundation for a discussion of temporal semantics in the preceding sections by introducing the concepts of different representations of time,

temporal instants and intervals, and (feature) states and events, this section now approaches an (aeronautical) event model.

Temporal Dimensions of Events

The preceding discussion of states and events has dealt with the temporal evolution of the modeled world, the *valid times* of feature states and events. Another dimension of time is to be considered when regarding data distribution processes. Under realistic conditions the notification of an event, sent from a Publisher to a User, does not reach the User system at the event time t_e, but at a later *notification time* $t_n > t_e$.

If an event happens at t_e, the validity of a state ends at that instant t_e. If that state had been valid since a former instant t_b, the validity interval of the state is $[t_b, t_e]$. The valid time of the state is independent of the notification time t_n.

Two individual dimensions of time have to be distinguished here, which shall be called the *valid time* dimension and the *dissemination time* dimension.[7]

Say, an event occurs at instant t_e. This instant is in the valid time dimension. Upon observation, the Publisher processes the event and sends out the event notification at publication time t_p. The notification is disseminated to all Users, which are notified of the event at notification time t_n. These times are in the dissemination time dimension. There is only one event valid time and publication time, but there may be many different notification times, because the former ones are Publisher-specific (of which there is only one for an event notification) and the latter ones are User-specific (of which there may be many for an event). Figure 3.7 illustrates the different event times.

The dissemination time instants t_p and t_n are important specifically for legal reasons where often (say, for the investigation of an air traffic accident) the question is what information a User had at a specific moment (say, the moment of the accident) rather than what was the actual state of the modeled world in the valid time dimension [68]. One might think that in an electronic communication systems environment, event notification is achieved in near real-time manner. Hence the time interval between t_e and t_n, when the event is observed, processed, published, disseminated and received is very small.

Considering end-to-end event notification from the Publisher to the Users of Aeronautical Information however, it is too idealistic a scenario to expect all affected Users being connected to the information dissemination system at the publication time t_p and thus being notified of the event in near real-time. The global distribution of Publishers

[7] In the Temporal Database research community, these two time dimensions have been given different names by different people in the last decades of research (see, e.g., [129, 159, 160, 119]). The consensus of today is “valid time” and “transaction time” [49, 46] with respect to temporal databases.

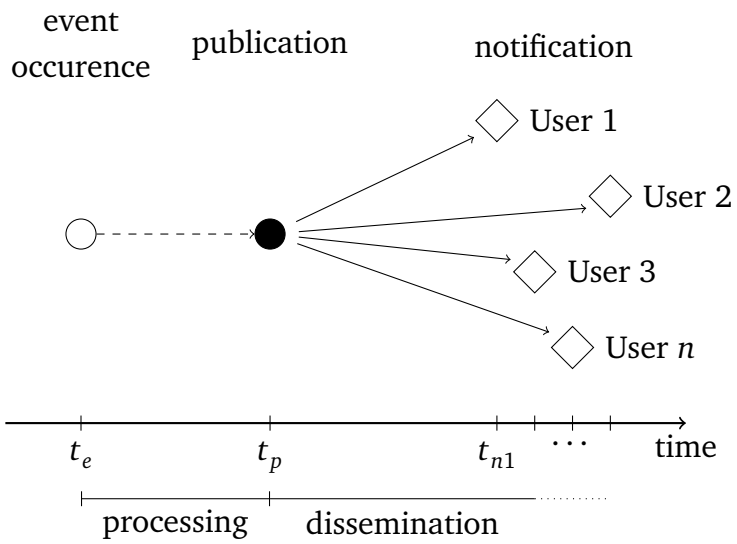


Figure 3.7.: Temporal dimensions valid time t_e, publication time t_p, and notification times $t_{n1}, \ldots$ of an (observed) event.

and Users and the highly heterogeneous system landscape connecting them are issues to be considered even when regarding a scenario in the far future. It must be expected that Users receive event notifications by various means such that the time needed for dissemination (and consequently the notification time) may be widely varying among Users.

The SESAR ATM Target Concept explicitly requires future systems and operations to account for stakeholders' heterogeneous technical equipment. In the case of aircraft for instance it may not be presumed that all aircraft are equipped with air-ground data links and appropriate airborne processing systems to receive event notifications during a flight in the future. This means in turn that event notifications must be received and appropriately processed before the flight. In general, event notification must be made *in advance* of the event's valid time.

Advance notification of events is possible because aeronautical events are planned changes to the aviation environment rather than "observed happenings of interest", which is the general assumption in the area of event-based systems [38, 138].

It is a crucial requirement for air traffic safety that all stakeholders are aware of the currently valid state of the aeronautical environment. Every stakeholder's notification instant t_n must therefore occur before the event instant t_e, $t_n < t_e$. Taking varying dissemination times for the individual Users into account, the event notification is published (in the best case) a relatively long time before the valid time. Figure 3.8 illustrates the order of occurence of publication time, notification time, and valid time of a planned aeronautical event.

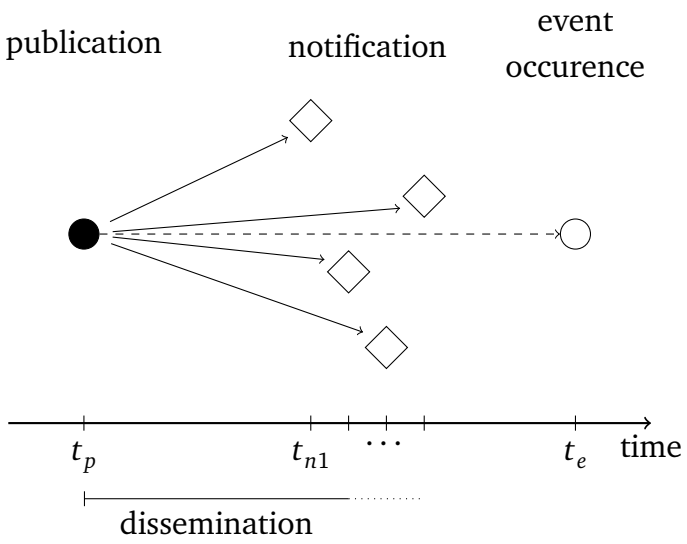


Figure 3.8.: Temporal dimensions publication Time t_{p}, Notification Times $t_{\mathrm{n1}}, \ldots$ and Valid Time t_{e} of a planned aeronautical event.

Coordinated Events

Regarding today's operations, the advance notification is especially important because Aeronautical Information is usually not processed by the end users directly, but by data integrator companies, who collect the information from AIS offices worldwide and produce "data products" like aeronautical charts or databases for the onboard Flight Management System (FMS) from it, which are then redistributed to end-user customers [74].

Additionally considering that paper products are a common means of information distribution (e.g. AIP) and presentation (e.g., aeronautical charts), the distribution process via paper products takes a long time. Furthermore, the production and distribution of data products requires additional time. Even in the case of electronic charts, safety considerations require that these charts are pre-composed and cannot be rendered on-the-fly, e.g., onboard the aircraft.

These processes have to be executed between the publication time t_{p} of an event notification and the event's valid time t_{e} to make sure that the event notification reaches the end-user in time.

With these time-intensive and costly processes, it is also desirable that the information in the end products remains valid for some time. Nowadays, this is achieved with the AIRAC process. It requires on one hand to publish event information long in advance of the event valid time so that data integrators receive the event information 28 days before its valid time. On the other hand, events are planned such that they do not occur

at arbitrary times but they are coordinated such that their valid time t_e coincides with the AIRAC effective dates (Figure 3.9).

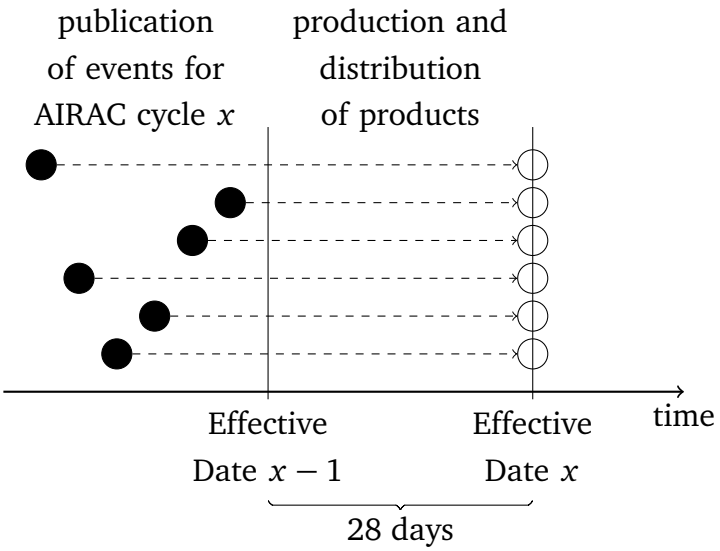


Figure 3.9.: AIRAC events

Considering the valid time dimension, Figure 3.10 illustrates AIRAC events affecting a feature instance. Events occur at the instants $t_0, \ldots, t_5$ exclusively. These are the AIRAC effective dates at a distance of 28 days. At t_0, the feature is created. Attribute value changes occur at t_1 and t_3. The feature is withdrawn at t_5.

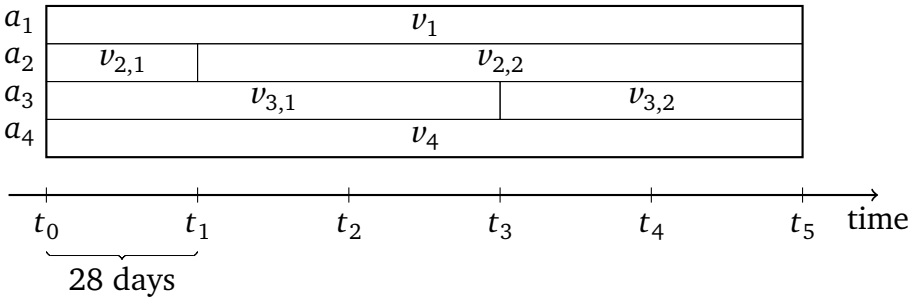


Figure 3.10.: AIRAC events and states in the valid time dimension

Temporary Events

Not all events can be coordinated within the AIRAC process. It may be necessary to publish an event notification on short notice, in which case the advance notification time required for production and distribution processes is not given. In addition, there is the notion of *temporary* Aeronautical Information events. Examples are the closure

of a runway for one day due to construction work or the erection of obstacles like construction cranes for a limited time span. It is obvious that in these cases it would not make sense to incorporate the events in products like aeronautical charts or FMS databases, because the information would become outdated during the long production and distribution process even before it reaches the end-user.

Hence, these events are handled separately in today's AIS operations, outside the AIRAC process.[8] There are two kinds of *temporary events*:

- A feature attribute is temporarily changed.
- A feature instance exists temporarily, i.e. it exists for a predefined time interval.

The first case is illustrated in Figure 3.11, where a temporary event overlays the regular state. Between t_{temp1} and t_{temp2}, the feature attribute a_1 has the value v_{temp}, thus overlaying the *baseline* value v_1.

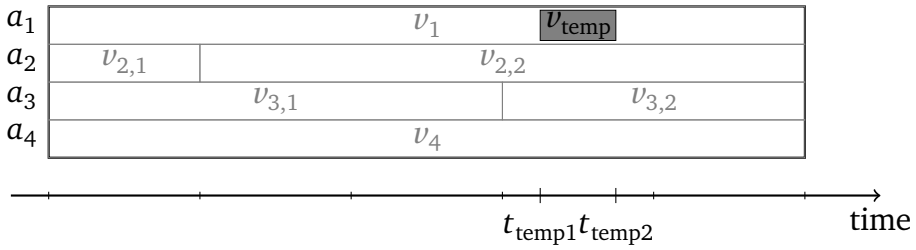


Figure 3.11.: Temporary event overlaying a "regular" feature state

It is easy to see that temporary events could be modeled just as well by means of two "regular" events. In the example, the same result could be achieved with two "regular" events, one at t_{temp1} changing a_1 from v_1 to v_{temp} and one at t_{temp2} changing a_1 back to v_1. The temporary event is hence an additional convenience construct. It is not necessarily required from a conceptual perspective, but it is required to support real-world conditions where event notifications cannot be distributed to all stakeholders in real-time. We shall come back to this point in the next section.

The notion of an event having a temporal extent is unusual. An event in the aeronautical world such as a change in a feature's state like the closure of a runway, may however in fact have a duration. Then the duration of the event, the runway closure, is the time period in which it persists, i.e., until the runway is reopened for operations. Such a notion of events with a duration is common in other research fields such as linguistics, as discussed later under "Related Work".

From a conceptual point of view, temporary events can just as well be seen as *temporary states*, because they (partially) describe (a part of) the state of the modeled world

[8] As described in 3.1.2, this means on a conceptual level to introduce another "information layer".

throughout a time period. From an implementation perspective, it makes however more sense to model this temporal concept as event, because an event's semantics of describing a state *transition* can be easily applied to describing a *deviation* from a state, thus representing merely a *delta*: the fact of the change.

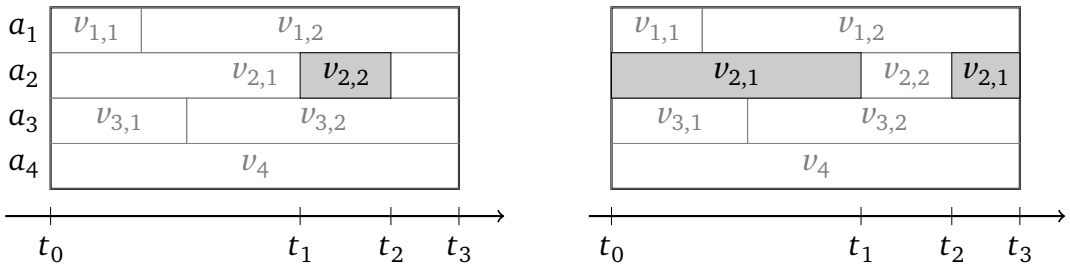
Throughout this thesis, the notion of instantaneous ("regular") and lasting ("temporary") events is retained. In order to distinguish the two classes of events, the former shall be called *permanent events* considering that the changes resulting from them are perceived as being permanent at the time of the event.

Baseline and Tempoary Layer

The layer on which permanent events apply is the *baseline layer*, and the one where temporary events apply is the *temporary layer*. States on the baseline layer are the previously defined "regular" states, which will in the following be referred to as *baseline states*.

While the notion of temporary events and states is not necessary from a conceptual point of view, it allows the discrimination of baseline and temporary states and in that way supports current operations involving the production and distribution of data products. Future operations may also benefit from the notion of temporary events. Temporarily restricting an airspace for specific flight operations or restricting an airport runway's approach procedure to specific aircraft categories because of forecast weather conditions are just two examples where the notion of temporary events is convenient to support ATM operations in the future as well.

The question arising with the discrimination of baseline and temporary states is: What makes a state baseline or temporary? Consider the example in Figure 3.12. The attribute a_2 has the value $v_{2,1}$ from t_0 until t_1 and from t_2 until t_3 and the value $v_{2,2}$ from t_2 until t_3. Both cases express exactly the same situation, just with one temporary event in 3.12a and two temporary events in 3.12b.



(a) Option 1 **(b)** Option 2

Figure 3.12.: Baseline and Temporary States

This example shows that the designation of a state as baseline or temporary is arbitrary and cannot be defined sufficiently within the model alone. Which way the situation would be expressed only depends on the definition of the "normal" circumstances which are modeled on the baseline layer. In Figure 3.12a, a_2's normal value is $v_{2,1}$ while in Figure 3.12a, it is $v_{2,2}$.

Baseline states express the "normal" situation and temporary states express the deviation from norm. The definition for "baseline" is thus based on the definition for "normal", which is arbitrary and has to be defined apriori based on operational requirements.[9]

Concluding this discussion of temporal semantics of (aeronautical) events, we are left with two notions of events: permanent events (happening instantaneously) and temporary events (having a duration). This distinction is required to support current and future AIS operations, where the distinction between two prioritized layers of information (baseline and temporary) is needed to allow for data products with long production times, which can incorporate baseline information only. This means that as long as data distribution processes do not happen in real-time between all involved stakeholders this distinction will be required, and is thus not dispensible even for a far away future.

Related Work and Conclusion

The notion of an event having a temporal extent is unusual. An event is most commonly thought of as having no (or very small) temporal extent. While this is the common notion in technical sciences (such as computer science in areas like event-based systems or active databases [33]), a durable event is in accordance with event semantics as discussed in other fields such as philosophy and linguistics (see, e.g., reference [93]). It is common sense for linguists that an event can be "*instantaneous*, such as two balls coming into contact, or *protracted*, such as the American presidential campaign" [148].

J. F. Allen has also contributed the interval notion of events to the general discussion of temporal semantics, see, e.g., his 1990 survey on ways to represent time and temporal dependencies [12] and his report on *Interval Temporal Logic* [13].

Event time represented by intervals has also been described previously in the context of event-based systems by C. Liebig *et al.* [124], but in the meaning of accuracy intervals describing temporal uncertainty, not in the meaning of event duration. A description of the duration of an event as temporal interval can be found in interval-based semantics for event detection in the area of active databases due to R. Adaikkalavan *et al.* [1,

[9] This however does not apply to temporary feature events. Temporary features such as a dynamically created restricted airspace exist for a limited time interval. If a feature instance exists on the baseline layer, it is not a useful notion to think of that feature as "temporarily not existing".

2]. There, so-called *composite events* consist of a number of *primitive events* that occur in some order. While primitive events are still assumed to happen instantaneously, the composite event may of course span over a temporal interval. It is then said that the composite event "occurs over an interval". Treating these composite events in the same semantics as primitive events, comparison expressions are required, for which J. F. Allen's (temporal) interval algebra is used, as we do in this work for interval filter operations (Chapter 4).

As a remark, it seems important to note that the introduction of planned temporary (lasting) and permanent (instantaneous) events must be regarded as an extension of the well-known model of observed, instantaneous events. It is not an alternative to it. In most other ATM information domains such as flight surveillance, for which the SWIM notification service shall be equally usable, the latter notion is more useful and common. The event model derived here seamlessly integrates with that notion, because exposing an event's valid time as a property for subscription is a concept that does not preclude other subscription models. In fact, for observed, instantaneous events, only rare cases may be imagined where the User of the notification service would subscribe for the event's valid time (which is, at the time of notification, in the past), because the User's temporal interval of interest is normally represented by the instants of subscription and unsubscription.

In conclusion, the valid time of an event may or may not be an aspect one might wish to subscribe to, and can or cannot be exposed for notification filtering, but it can (and will in most applications) be used together with other subscription models.

The remaining challenge is however to implement an event notification representation equally valid for permanent and temporary events, where the valid time is exposed as instant and temporal interval, respectively. The implementation we use for the aeronautical event notification service is presented later in Section 3.4 after discussing the spatial model required for aeronautical information in the next section.

3.3 Spatial Model

ATM-related spatiality is naturally Earth-related, i.e., a spatial model for the ATM Information Reference Model must be a *geographic* (or, *geospatial*) data model. Earth-related space is commonly regarded separately in the *horizontal* and *vertical* dimension, because different reference and coordinate systems are used: Whereas the horizontal position is specified in geographic coordinates, latitude and longitude, with respect to some geographic coordinate system, the vertical position, height, is specified as the distance from some reference such as ground or mean sea level (MSL).

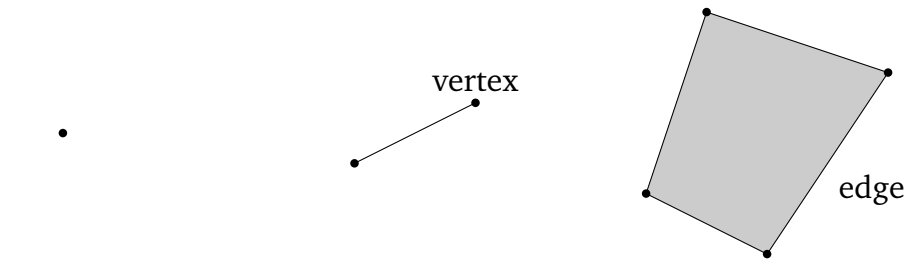
Horizontal and vertical space is therefore introduced and discussed subsequently in the following. Subsection 3.3.1 first introduces basic 2-dimensional spatial datatypes and the spatial profile of ISO standard 19107 implemented in AIXM 5 as well as the specifics of geospatial data.

Different path types are used in aerial navigation leading to different notions of "straight" lines, which impose specific requirements on the horizontal spatial model. This is discussed in Subsection 3.3.2 and the resulting horizontal spatial model as implemented in AIXM 5 is briefly presented in Subsection 3.3.3. The third spatial dimension is discussed with respect to geodetic datums in Subsection 3.3.4, and the resulting "2.5-dimensional" spatial model for Aeronautical Information is presented in Subsection 3.3.5.

3.3.1 Fundamentals of (Geo-)Spatial Data

A feature's spatial attributes like extent or location take the form of spatial objects. An often used formal framework for the description of spatial objects are point sets, where space is regarded as an infinite set of dimensionless points. A point is identified by its coordinates (in 2-dimensional Euclidean space elements of $\mathbb{R} \times \mathbb{R}$). Higher-dimensional objects, i.e. lines, polygons, polyhedra, are infinite point sets over this space, which are specified as entities by their bounding lower-dimensional elements. Lines are bounded by *vertices*, polygons are bounded by *edges*, polyhedra are bounded by *faces* [153].

The common represenation of these objects is *boundary representation (BREP)* [114], where a spatial object is described by its boundary through different basic *geometries*: points, lines, and polygons. Points, lines, and polygons are the coordinate representations of vertices, edges, and faces, respectively, and build the set of basic *spatial datatypes*. Figure 3.13 illustrates the three basic spatial data types.

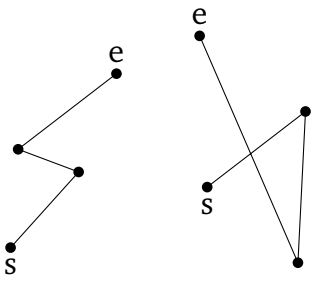


(a) Point **(b)** Line **(c)** Polygon

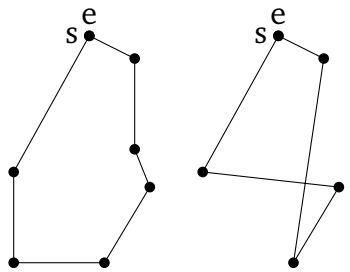
Figure 3.13.: Basic spatial datatypes: point, line, polygon

In 2-dimensional space, a point P would thus be represented as a coordinate pair, $p = (x, y)$. A line ℓ is described by its two bounding vertices $\ell = \big((x_1, y_1), (x_2, y_2)\big)$. A polygon P is represented by its surrounding boundary line in the form of a sequence of consecutive edges, a *linestring* $\ell s = \big((x_1, y_1), \ldots, (x_n, y_n)\big)$. The consecutive edges are linked at the begin and end vertices. The last vertex is the same point as the first, which gives a *linear ring*, $\ell r = \big((x_1, y_1), \ldots, (x_n, y_n)\big)$, where $x_1 = x_n, y_1 = y_n$.

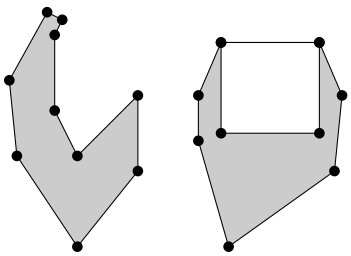
The *Simple Feature Profile* of ISO standard 19107 [142] implemented in GML [143] and consequentially in AIXM 5 [26] allows these basic 2-dimensional spatial datatypes only. It additionally requires all geometries to be "simple" (Figure 3.14): A linestring is considered simple if it does not cross itself, i.e., there are no two edges in the linestring that cross (a). A simple linear ring is a simple linestring with coinciding begin and end vertices (b). A polygon is considered simple if its outer boundary is represented by a simple linear ring and it has no holes or punctures, its interior thus forming a connected set (c).



(a) Simple and non-simple linestring



(b) Simple and non-simple linear ring



(c) Simple and non-simple polygon

Figure 3.14.: Simple and non-simple geometries

While spatiality is most often regarded in Euclidean space because of its well-known metric properties and simple computations, geographic (or *geospatial*) data are the special case of spatial data where spatiality is described with respect to the Earth's surface [169]. Regarding two-dimensional represenations on the Earth's surface, the embedding space is thus not Euclidean.

A point on the surface of the Earth is usually described in *geographic coordinates* using a spherical coordinate system. The origin of the coordinate system is the center of the Earth, and a point is described by angles measured at the origin. The *latitude* ϕ of a point denotes the angle between the point and the equatorial plane. The *longitude* λ is the angle east or west of the north–south line between the two geographical poles, that passes through Greenwich in the UK, which is an internationally agreed convention[10]. Lines of constant latitude are called *parallels*, lines of constant longitude *meridians*. The zero-longitude line is called the *Prime Meridian*. A convention is that the longitude is negative west of the Prime Meridian, and latitude is negative south of the equator (Figure 3.15). The domain of the angles is $-90° \leq \phi \leq +90°$ and $-180° \leq \lambda \leq 180°$[11]. A point P on the Earth's surface is uniquely specified as $P = (\phi_P, \lambda_P)$.

A point on the Earth's surface can naturally be described in Cartesian coordinates in $\mathbb{R}^3$. Given a right-handed coordinate system such that the origin is at the Earth's center, the positive z-axis cuts through the north pole, and the x-axis and y-axis span the equatorial plane, a point P at latitude ϕ and longitude λ is given by

$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \cdot \begin{bmatrix} \cos\lambda\cos\phi \\ \sin\lambda\cos\phi \\ \sin\phi \end{bmatrix}, \tag{3.1}$$

where R is the distance from the origin, the Earth's center. Assuming a spherical Earth model[12], R is a constant, the radius of the Earth.

[10] In fact, the internationally agreed zero-longitude line passes near the (former) Royal Observatory at Greenwich.

[11] In the following, we will use the degree notation common with geographic coordinates in the text. Where they are later in Section 4.3 used as angles in trigonometric equations, radians are assumed with $-\pi/2 \leq \phi \leq +\pi/2$ and $-\pi \leq \lambda \leq +\pi$.

[12] Different Earth models will be discussed in Section 3.3.4. Assuming spherical geometry can produce errors up to 0.55 % at the equator, though generally below 0.3 %, depending on latitude and direction [178].

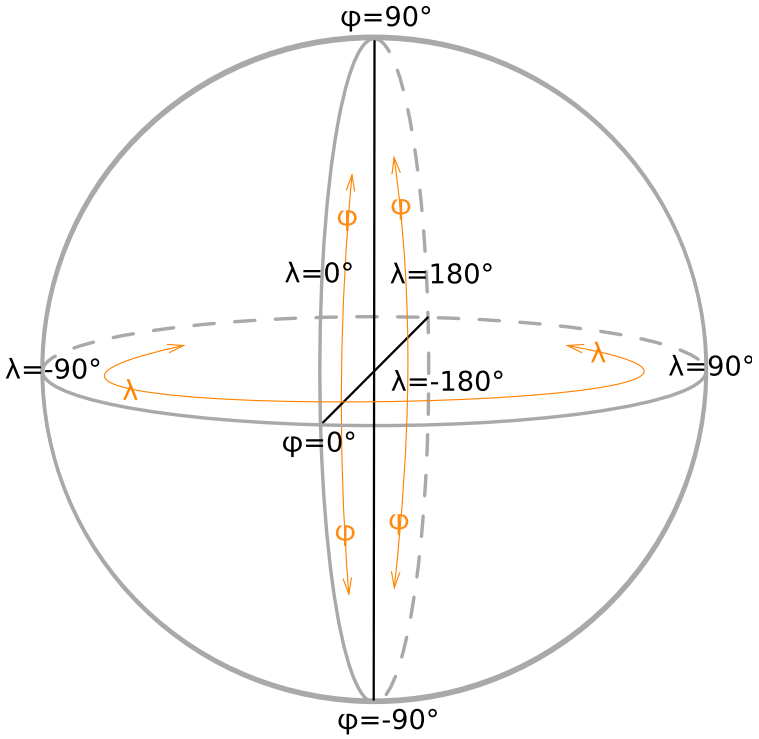


Figure 3.15.: Geographic coordinate system with latitude ϕ and longitude λ (from [173]).

Inverting Equation (3.1) and normalizing appropriately, the geographic coordinates are derived from the Cartesian coordinates by

$$\begin{aligned} \phi &= \arcsin(z) \\ \lambda &= \operatorname{arctan2}(y, x), \end{aligned} \tag{3.2}$$

where *arctan2* is an extension of the conventional *arctan* function taking the input's quadrant into account. It is usually defined as

$$\operatorname{arctan2}(y,x) = \begin{cases} \arctan\frac{y}{x} & \text{if } x > 0 \\ \arctan\frac{y}{x} + \pi & \text{if } x < 0, y \geq 0 \\ \arctan\frac{y}{x} - \pi & \text{if } x < 0, y < 0 \\ +\frac{\pi}{2} & \text{if } x = 0, y > 0 \\ -\frac{\pi}{2} & \text{if } x = 0, y < 0 \\ 0 & \text{if } x = 0, y = 0 \end{cases}$$

3.3.2 Aerial Navigation Paths

A specific issue with geographic coordinates comes with paths used in aerial navigation: The boundary representation model assumes that an edge is straight, i.e., it is uniquely defined by its two end points P, Q through linear interpolation. In $\mathbb{R}^2$, such a line takes the form of the shortest Euclidean distance. The shortest Euclidean distance between two points $P = (\phi_P, \lambda_P), Q = (\phi_Q, \lambda_Q)$ on the Earth's surface, however, always cuts through the Earth.

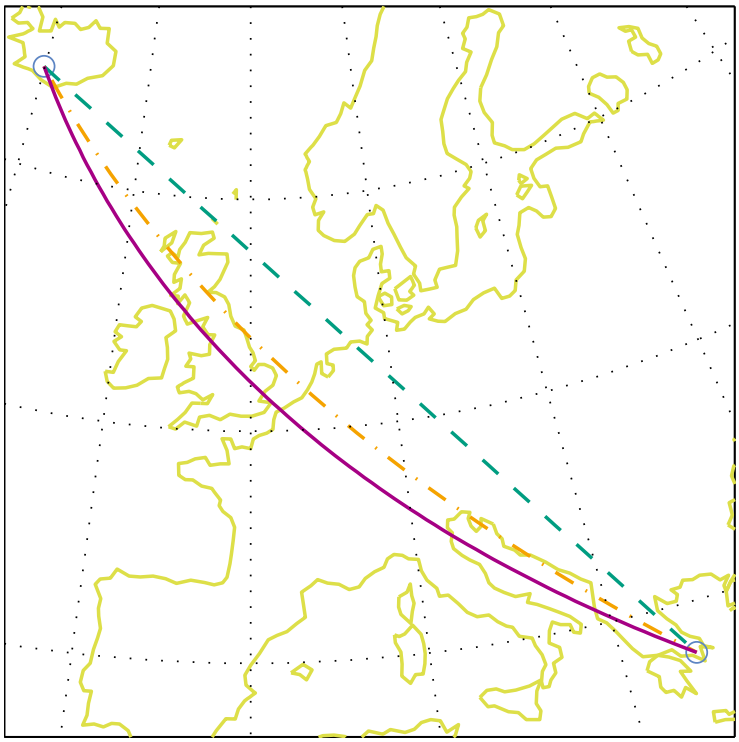
A line on the Earth's surface can only be straight in some projection to a 2-dimensional Cartesian coordinate system, a *map projection*. No such projection can be free of distortion [176]. Many different map projections have been proposed and used in history and still are. Depending on the application, projections can be chosen such that they are conformal (locally preserving angles), equal-area (preserving areas), equidistant (preserving distance from some standard point or line), or a compromise of these, but never all at the same time [161]. This leaves us with different notions of "straight" lines, depending on the projection that displays a given line on the Earth's surface as straight. Three types of lines are specifically important in aviation:

Orthodrome (or *great circle route*) An orthodrome (from Greek *ortho*: straight + *dromos*: running) is the shortest path between two points on the Earth's surface. It is a segment of a *great circle*. A great circle is a circle on the surface of a sphere that has the same circumference as the sphere. It is uniquely defined by two points P, Q that lie on the great circle. The orthodrome is the shorter one of the two great circle distances. The only map projection that displays great circles as straight lines is the Gnomonic projection (Figure 3.16a).[13]

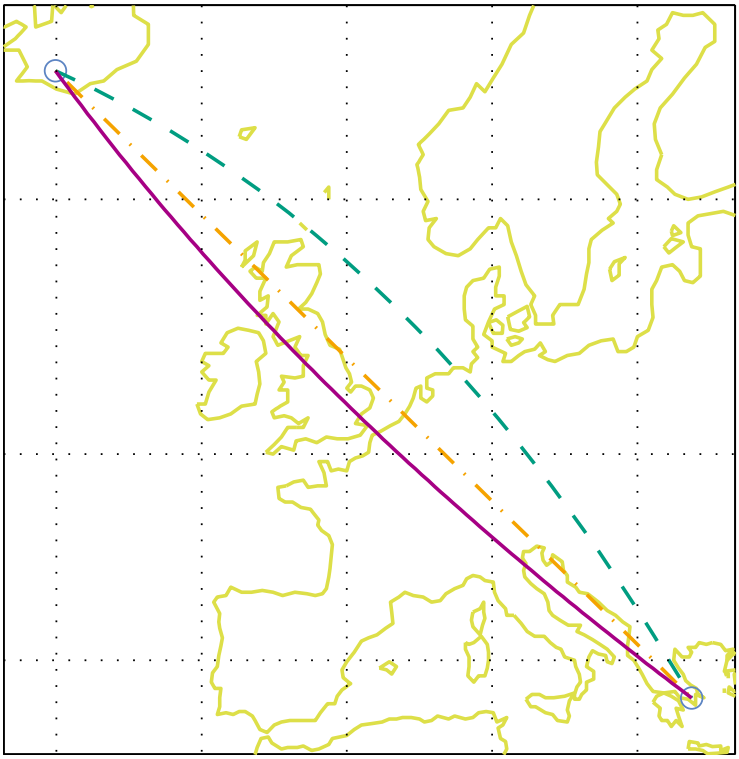
Loxodrome (or *rhumb line*) A *loxodrome* (from Greek *loxos*: oblique + *dromos*: running) is a path following a constant bearing, i.e., it crosses all meridians at the same angle. It has a long history in navigation, because it is much easier to follow than the shorter orthodrome, which requires to continuously adjust the course. For its prevalence in navigation, the Mercator projection is a very commonly used and familiar map projection, because loxodromes are straight lines in such a projection (Figure 3.16b).

Linear interpolation In a linear interpolation of latitude ϕ and longitude λ, the coordinates (ϕ, λ) are treated as if they were Cartesian coordinates in $\mathbb{R}^2$. The equivalent map projection, where $x_{\text{map}} = \lambda, y_{\text{map}} = \phi$ is the *Equirectangular projection* with

[13] The Lambert conformal conic projection is often used for aeronautical charts, because straight lines on such a map approximate orthodromes. It has the advantage over the gnomonic projection that it does not distort shapes as much.

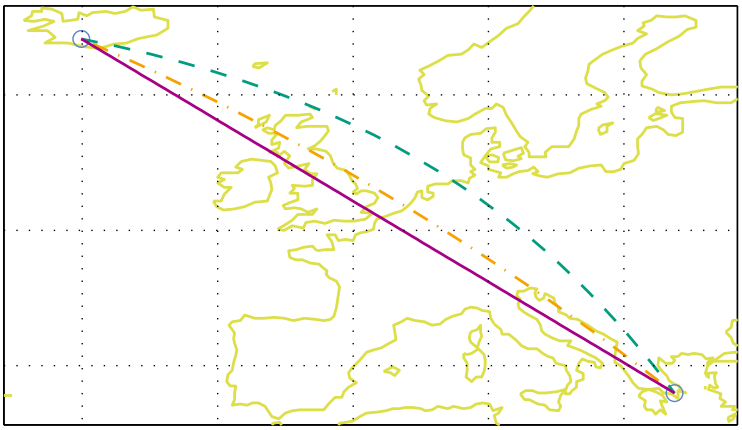


(a) Gnomonic projection



(b) Mercator projection

Figure 3.16.: Linear interpolation (solid), loxodrome (dash-dotted) and orthodrome (dashed) between Reykjavík and Athens in different map projections



(c) Plate Carée

Figure 3.16.: Linear interpolation (solid), loxodrome (dash-dotted) and orthodrome (dashed) between Reykjavík and Athens in different map projections

the equator as the standard parallel, which is also known as the *Plate Carée* (Figure 3.16c).

3.3.3 Horizontal Aeronautical Spatial Model

The Simple Features Profile only supports linear interpolation, while loxodromes and orthodromes are obviously no linear interpolations between latitude/longitude points. Since these path types are however crucially required in aeronautical information, the ISO standard 19107 and GML profiles implemented in AIXM 5 were extended such that other curve interpolation types are permissible for the description of edges [26]. The GML standard also had to be changed to allow the specific interpolation types “geodesic” and “rhumb line”, for which the author of this thesis submitted a Change Request to the OGC [84].

Hence, the type of interpolation must be specified explicitly when defining lines, linestrings or polygon boundaries [25]. Figure 3.17 shows a simplified[14] representation of the resulting 2-dimensional (*horizontal*) aeronautical spatial model.

The non-linear interpolation of lines has implications for the geometric algorithms used in the event nofitication service for subscription handling and notification matching, which will be discussed later in Section 4.3.

[14] In this representation, it would be possible to have different horizontal datums (described next) for the end vertices of a line, which is in fact not allowed, and a set of arbitrary lines as polygon boundary disregarding that polygon boundary edges have to form a linear ring.

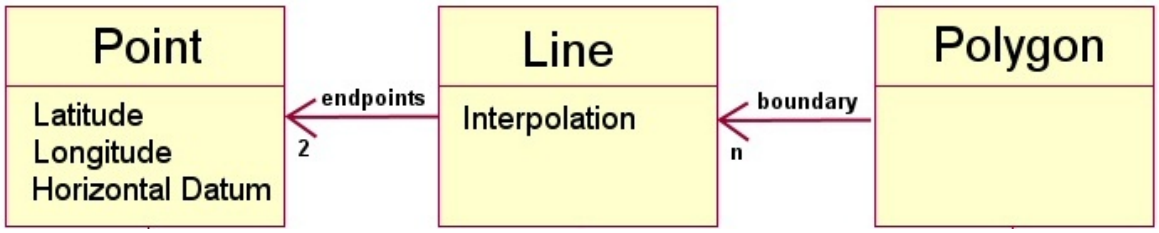


Figure 3.17.: Geospatial objects `Point`, `Line`, `Polygon` in their aeronautical implementation with explicit specification of the line interpolation type

3.3.4 Geodetic Datums and the Third Dimension

Latitude and longitude merely describe a point on the Earth's surface. Obviously, in an aeronautical spatial model, means are required to specify *height*: locations over the Earth's surface (*altitude*) as well as locations on the irregularly changing topographic surface (*elevation*). Height is commonly specified as distance from mean sea level (MSL).

MSL is a description for the imaginary ocean surfaces if they were in rest and connected. It is the equipotential surface of the Earth, which is everywhere perpendicular to the direction of gravity (Figure 3.18) [103] and is described in a *vertical datum*, a *geoid* model.

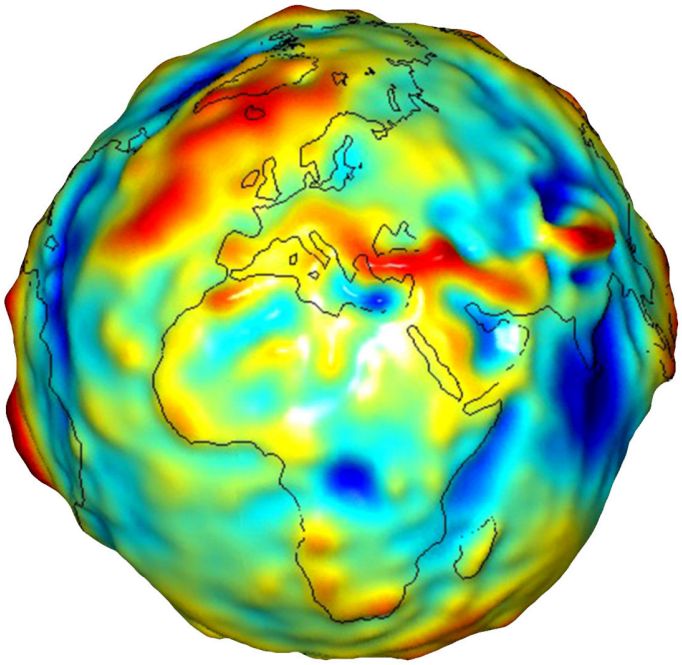


Figure 3.18.: Rendering of the Earth's Gravity Field (from [73])

Since a geoid's mathematical description is very complex due to its irregular shape, the Earth's shape is often approximated by simpler Earth models. The simplest type of model is the spherical Earth model, which has been assumed in the discussion so far. In this model, the radius R of the Earth is constant, and different spherical Earth models vary in R's value. Nowadays 6,371 km is commonly assumed as an average of the real value based on centuries of survey and decades of satellite measurements.

A better, somewhat more complex, approximation of the Earth's shape is an oblate spheroid or biaxial ellipsoid. Ellipsoidal Earth models vary in the lengths of the semi-major axis a and the semi-minor axis b, often specified as a single value in the *flattening* $f = \frac{a-b}{a}$.[15]

The common Earth model in aviation and other global applications today is the World Geodetic System 1984 (WGS-84) [139, 96, 60], which defines the ellipsoid values: $a = 6,378,137$ m at the equator, $b = 6,356,752.314245$ m at the poles. This gives a flattening of $f = 1/298.257223563 \approx 0.335\%$. The Earth's radius varies in this model from approximately 6,336 km (equatorial meridian) to 6,399 km (polar).

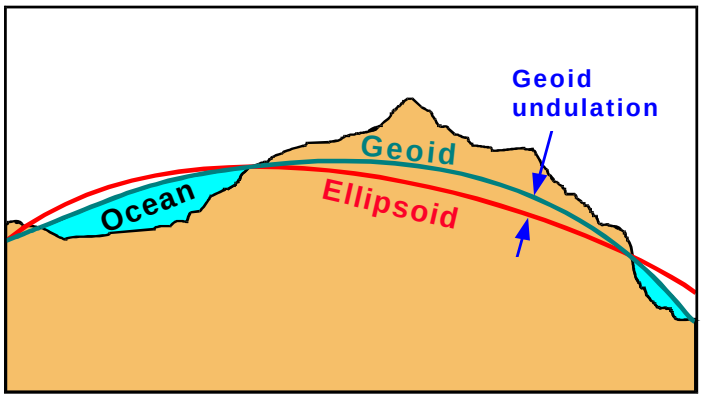
Based on such an Earth model, the geoid is simply described as the deviation from the reference ellipsoid, the *geoid undulation* (Figure 3.19a). For any point $P = (\phi_p, \lambda_p)$, the exact geoid undulation value can be specified. Figure 3.19b shows a rendering of the geoid undulation in WGS-84 for the whole Earth. The geoid in WGS-84's vertical datum is specified through the Earth Gravitational Model 1996 (EGM-96) [120]. It is also the global aviation standard geoid model recommended by ICAO [98, 96].

Given a complete geodetic datum, i.e., reference ellipsoid and geoid in horizontal and vertical datum, respectively, the height h at some point is thus simply described as the distance from the geoid at this point.

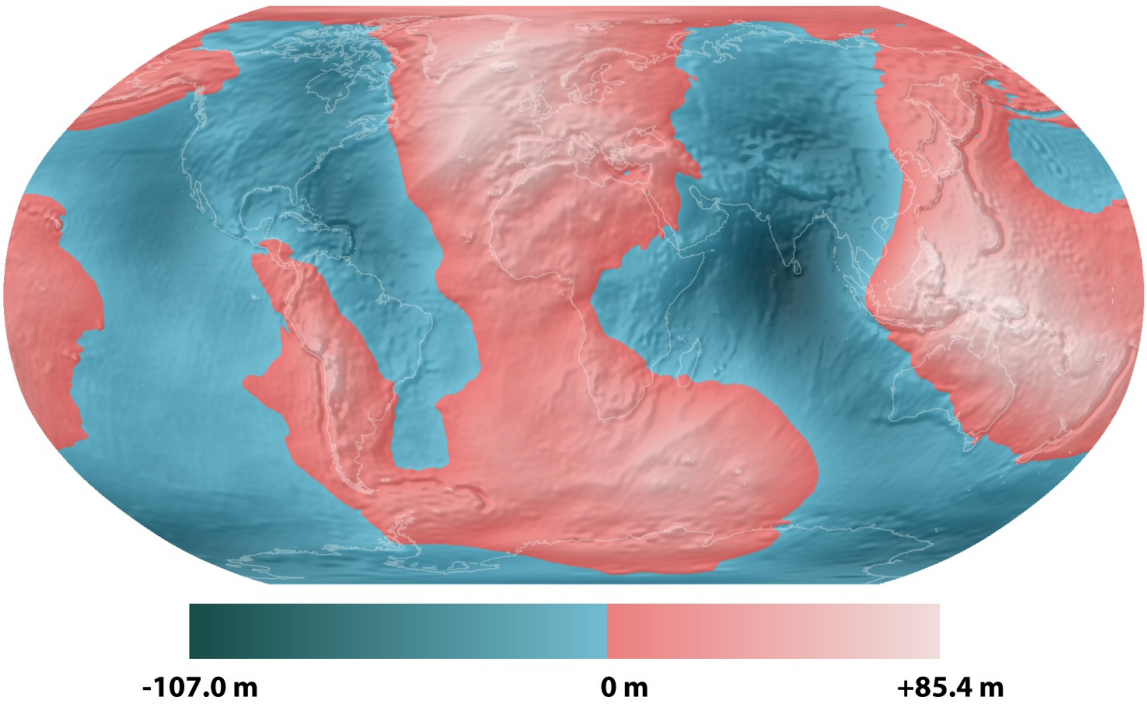
In aerial navigation, horizontal and vertical positioning are achieved by different means. Altitude is measured by air pressure (enroute) or radar (in terminal airspace), and the determination of the horizontal position has traditionally relied on navigation facilities installed on the ground[16]. Satellite navigation also plays an important and growing role in air navigation through the installation of Global Navigation Satellite Systems (GNSSs): the United States NAVSTAR-GPS, widely known as Global Positioning System (GPS), the Russian GLONASS, and (in future) the European GALILEO [95]. While these systems are generally capable of determining latitude, longitude, and altitude, the latter is referenced to the respective ellipsoid model, which may deviate from the geoid (through geoid undulation) and consequently from the altitude value measured barometrically.

[15] A comprehensive overview of different Earth models is given in reference [45].

[16] Navigational aids such as Distance Measuring Equipment (DME) or VHF Omnidirectional Range (VOR) allow the determination of distance and/or angle to the facility, the exact location of which (as well as frequency and other parameters necessary for use) is published through AIS.



(a) Ellipsoid, geoid and topographic surface



(b) Difference between the EGM-96 geoid and the WGS-84 reference ellipsoid (from [174])

Figure 3.19.: Geoid undulation

The altitude of aircraft in the enroute phase of a flight (outside terminal airspace, between departure and arrival procedures) is defined by atmospheric pressure and is specified in *flight levels*. A flight level (FL) altitude is calculated from a world-wide fixed pressure datum of 1013.25 hPa, where one flight level equals 100 ft. Depending on the actual local air pressure, which varies through meteorological conditions, one FL does not necessarily equal 100 ft and is therefore not the aircraft's true altitude. The deviation of flight levels from the true altitude is however not an issue for conflict-free airspace organization, because all aircraft measure altitude barometrically enroute and altitudes are assigned by ATC in flight levels.

3.3.5 2.5-dimensional Aeronautical Spatial Model

The different horizontal and vertical datums as well as the different means of navigation are the reasons for space being regarded separately in the horizontal and the vertical dimensions. Horizontally, a point is described by its latitude/longitude coordinate with respect to, e.g., WGS-84, $P = (\phi, \lambda)$. The vertical location of an elevated point is specified in its elevation e as the distance from some geoid, e.g., EGM-96. Hence, a point spatial object is given by $P = (\phi, \lambda, e)$. The horizontal and vertical reference systems must however be explicitly specified or implicitly assumed.

The Aeronautical Information spatial model is a "2.5-dimensional" model: Lines and polygons are not built on elevated points, but on points in the 2-dimensional horizontal reference system, and add an elevation value as required to form elevated spatial objects. Thus, lines or polygon boundary edges cannot be made up of points with different elevations, but the simple spatial object has one elevation.

Additionally, the interpolation method used for lines or polygon boundary edges is to be specified.[17] Figure 3.20 shows the resulting spatial objects.

3.4 Aeronautical Event Notifications

Based on the preceding discussions of temporal semantics of aeronautical events and the geospatial model for aeronautical data, we can now define the aeronautical event notification model. We model notifications thus that they abstract the event information with the event's *spatiotemporal effectivity*.

[17] Furthermore, Annex 15 requires to explicitly specify the geoid undulation and the vertical datum used wherever a height value at geographic coordinates is specified. The use of WGS-84 as geodetic system is generally obligated by ICAO and its ellipsoid is implicitly assumed as horizontal reference system, but heterogeneous vertical reference systems are still in common use and must therefore be specified explicitly.

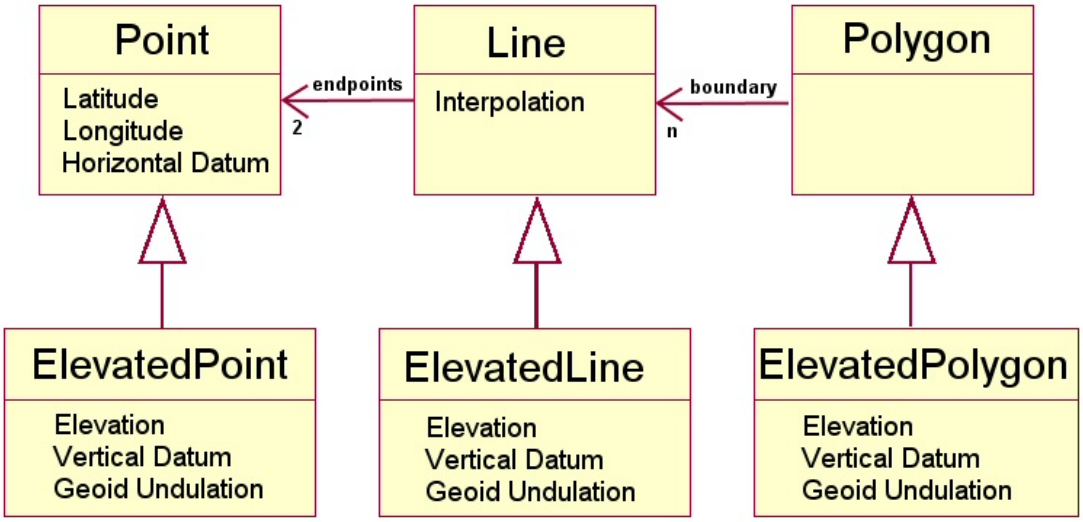


Figure 3.20.: Geospatial objects of an aeronautical 2.5-dimensional model: `Point`, `Line`, `Polygon` in two horizontal dimensions, and their elevated extensions.

For instance, the notification for an aeronautical event affecting the Upper Flight Information Region (UIR) Rhein airspace (Figure 3.21) in the time from 1 pm until 2 pm and at the flight levels 250 up to 300 is represented in the filter space as depicted in Figure 3.22.

We implement an aeronautical event's spatiotemporal effectivity as a tuple (T, H, V), where T denotes the event's valid time, and H and V denote the horizontal and vertical space the event affects. An aeronautical event notification n is thus modeled for our purpose simply as $n = (T, H, V)$. The implementation of the individual parts is discussed in the following subsections.

3.4.1 Temporal Effectivity

Three different event times must be distinguished: the valid time t_e, the publication time t_p and the notification time t_n. Out of these times, it is the event's valid time only that affects the modeled world, and is therefore the time that a User that subscribes to event notifications is interested in. For instance, when planning a flight that is supposed to take place in the time $[t_{takeoff}, t_{touchdown}]$, any event with valid time $t_{takeoff} \leq t_e \leq t_{touchdown}$ could possibly affect the flight.[18] This means that Users of the SWIM pub/sub system must be provided with the means to subscribe to events based on their valid time and in turn, event notifications must expose the event's valid time to be matched by User subscriptions.

[18] This assumes that t_e is an instant. If it were an interval, it could affect the flight if it *intersected* with the flight time. This is discussed later in Section 4.1.

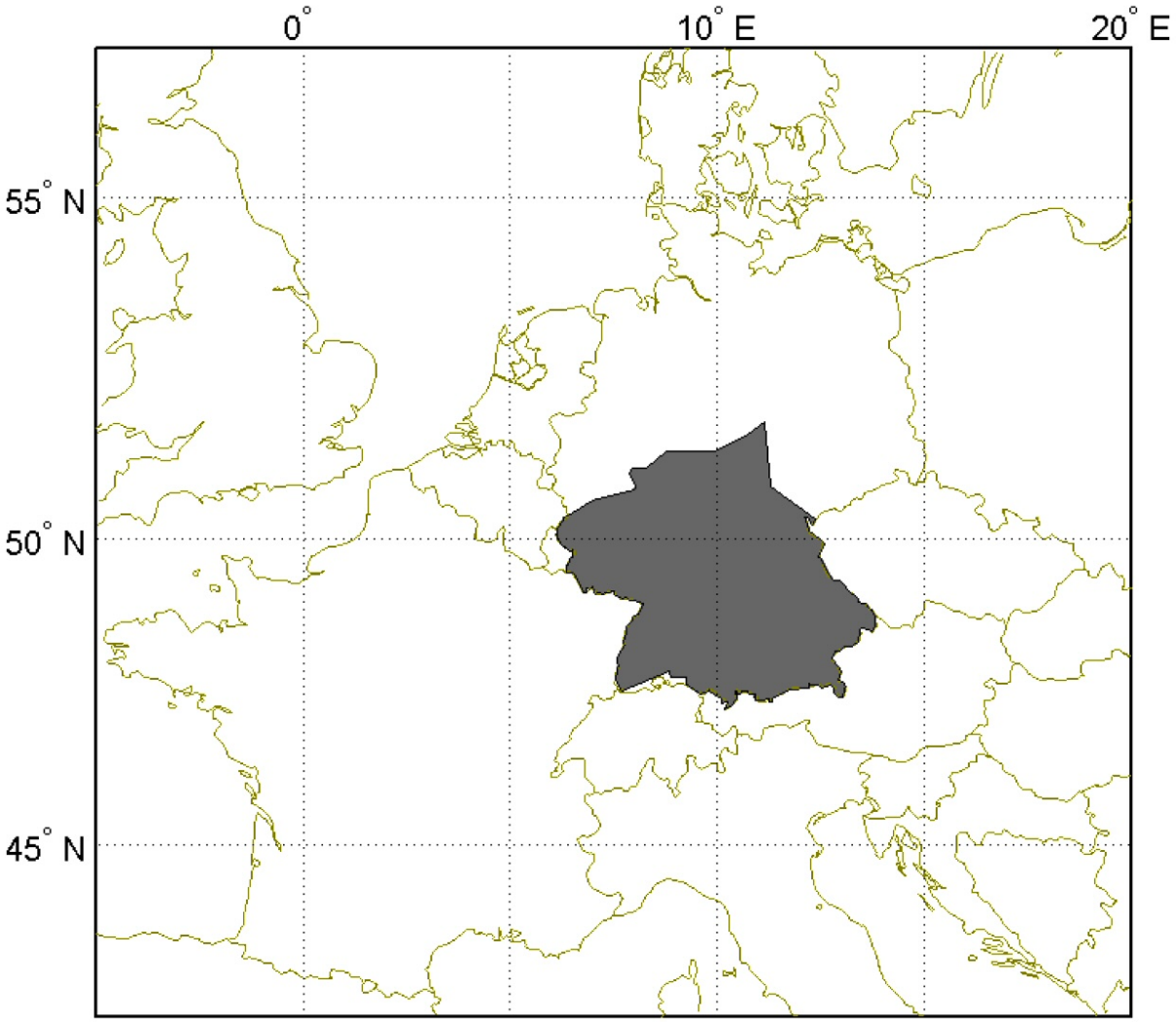
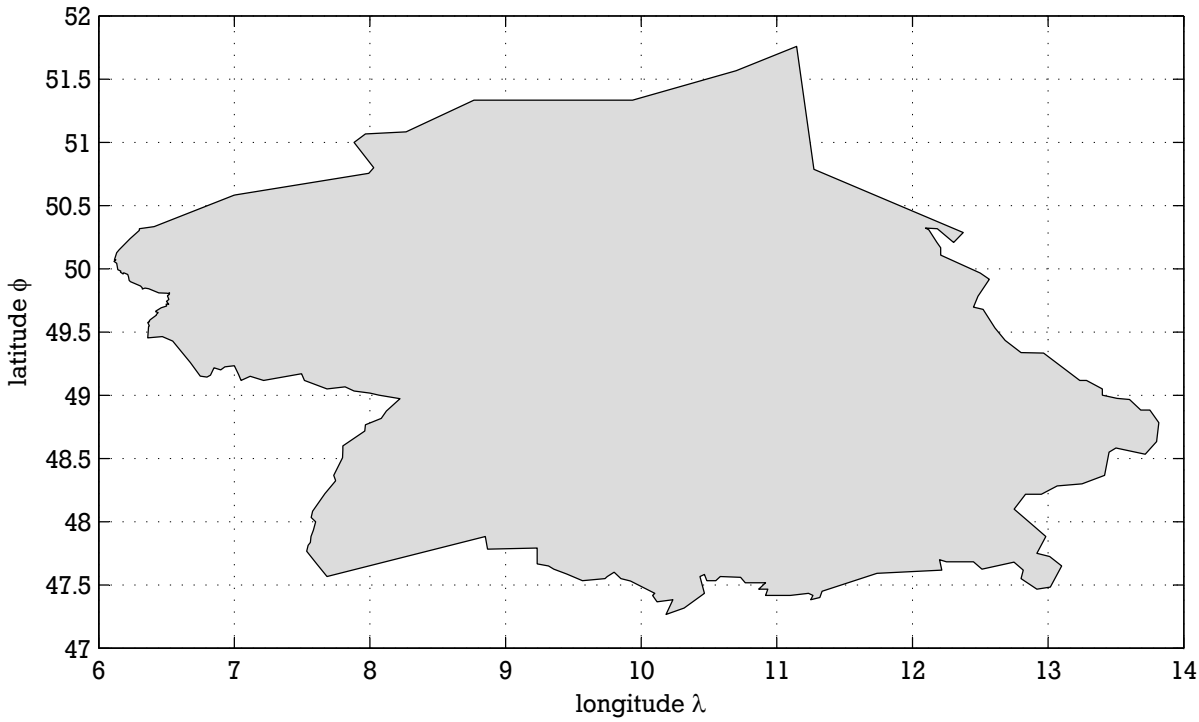
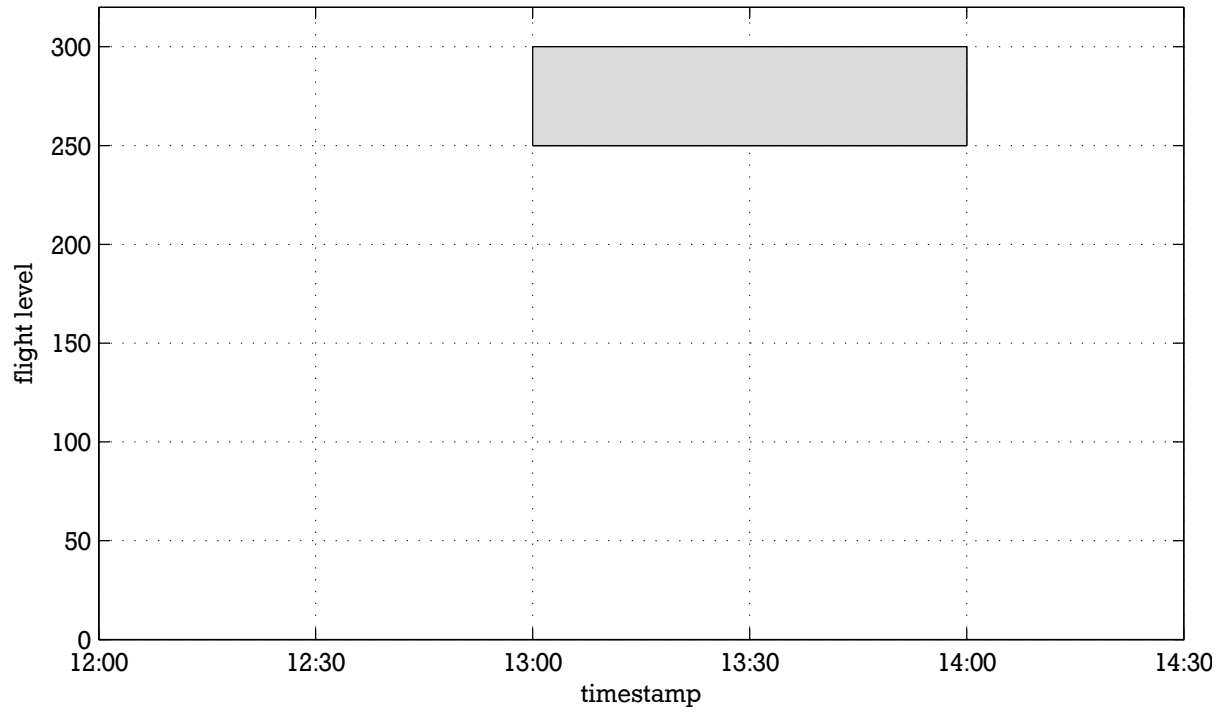


Figure 3.21.: Upper Flight Information Region “Rhein”



(a) longitude and latitude



(b) time and altitude

Figure 3.22.: Notification of an event affecting Rhein UIR

We therefore model aeronautical event notifications such that they have an attribute T denoting the event's *temporal effectivity*, which is the event's valid time. We apply a discrete time model, where the duration of a chronon is defined systemwide. Instants are time points in the resolution of a chronon, and temporal intervals are regarded as sets of consecutive time points. In addition, intervals are generally right-open (for the reasons mentioned in Section 3.2.1).

A permanent event E^{t_e} is valid at an instant, hence $T = t_e$, whereas a temporary event $E^{\left[t_{\text{begin}}, t_{\text{end}}\right)}$ is valid throughout a time interval, hence $T = \left[t_{\text{begin}}, t_{\text{end}}\right)$. The discrete time model allows to consistently write the permanent event's temporal effectivity T as interval also, $T = \left[t_e, t_e + 1\right)$, due to the unique definition of a successor "+1". This interval includes one single time point, the event's valid time t_e. This way, T is consistently implemented as a right-open interval of time points.

As an example, the notification of the above temporary event with valid time from March 14, 2008, 1 pm, until March 14, 2008, 2 pm, exposes the event's temporal effectivity in the attribute

$$T = \left[\texttt{2008-03-14 13:00:00}, \texttt{2008-03-14 14:00:00}\right)$$

assuming one second as chronon.

3.4.2 Horizontal Spatial Effectivity

The spatial effectivity of aeronautical events is separated into the affected horizontal area (the "section of the surface of the Earth") and the affected height (the "vertical extrusion" of this section).

The event's *horizontal spatial effectivity* is modeled in the notification attribute H. We implement H as a polygon such that the space affected by an event always has an extent, because even if the aeronautical event affected for instance a feature at a point location like a navigational aid, the effect of the change to it reaches into space in all dimensions. The horizonal spatial effectivity H is therefore implemented as a polygon.

In the case of the above example event for instance, the UIR Rhein is defined in the AIP of Germany as a polygon the boundary of which is partly specified by WGS-84 latitude/longitude points connected by orthodromes, and partly through the country's borders with neighboring countries, parts of which are specified as loxodromes [8]. The horizontal spatial effectivity of the event is therefore

$$H = \Big(\Big(\big((50°20'\mathrm{N}, 6°24'30''\mathrm{E})(50°35'\mathrm{N}, 7°\mathrm{E})\big), \texttt{orthodrome} \Big), \\ \Big(\big((50°35'\mathrm{N}, 7°\mathrm{E})(50°45'20''\mathrm{N}, 7°59'30''\mathrm{E})\big), \texttt{orthodrome} \Big), \\ \ldots \\ \Big(\big((\ldots)(\ldots)\big), \texttt{loxodrome} \Big), \cdots \Big)$$

3.4.3 Vertical Spatial Effectivity

Regarding the vertical spatial dimension, the consideration about extent applies here also. Hence, we implement the vertical space that is affected by the event (its *vertical spatial effectivity*) in a notification attribute V that is an interval specified by two height values, $V = \left[v_{\text{lower}}, v_{\text{upper}}\right]$.

For the sake of consistency, the notion of intervals over discrete points as used for temporal intervals is applied here, too. Generic intervals in this sense can be built over abstract point types in analogy to a discrete time model [127, 46]. If a successor function "+1" is defined uniquely for a point type $\mathbb{P}$, an interval I over this type, $I = \left[p^-, p^+\right]; p^-, p^+ \in \mathbb{P}$, represents the enumeration of all points from p^- to p^+, i.e., the set of all points encountered by succesively applying the successor function on the first point p^- until arriving at the last point p^+, $I = \{p^-, p^- + 1, p^- + 1 + 1, \ldots, p^+\}$.[19]

A point type for height is obtained by defining (in analogy to a chronon) a system-wide minimum height "step", e.g. 1 meter, 10s of feet, or 1 flight level[20], which quantizes the height dimension. Only multiples of this height quantum are allowed as height values. Altitude intervals can thus be treated like temporal intervals over time points.

In line with temporal intervals, we also use right-open intervals only. As an example, an event affecting the flight levels 245 up to and including 299 (assuming one flight level as the height quantum), the event's vertical spatial effectivity would be specified as

$$V = \left[245, 300\right).$$

[19] The existence of a unique successor function is not the only requirement for a type $\mathbb{P}$ to qualify as a point type, but an in-depth discussion is out of scope here. It can be found in reference [46].

[20] With respect to the latter, the discrete model of heights is quite intuitive. Since flight levels are assigned and not measured, they are commonly perceived as discrete. Fractions of flight levels (e.g., "flight level 123.45") are not assigned.

4 Spatial and Temporal Subscription Filters

In this chapter, the subscription model of the aeronautical event service is presented. It enables the Users to express their interest in time and space through *interval filters* and *spatial filters*. These basic filter types are formally introduced in Sections 4.1 and 4.2, respectively, where a definition is given, and the conditions for notification matching and filter relationships required by filter handling optimizations are described. The specific implementation issues for geospatial filters arising with different path interpolation types are discussed in Section 4.3.

Spatiotemporal filters are required to subscribe for aeronautical event notifications as introduced in the previous chapter. They are conjunctions of interval and spatial filters, which allow to subscribe to a temporal interval, a horizontal region, and an altitude interval. They are presented finally in Section 4.4.

4.1 Interval Filters

Assume we had simple notifications that expose the event's content as an interval $I_E = \left[i_E^-, i_E^+\right)$. That could for instance be the event's temporal effectivity if this were the only event characteristic Users can subscribe to. Then the notification is abstracted to $n = \left(I_E\right)$. Subscriptions for such interval notifications take the form of *interval filters*. An interval filter $F = \left(I_F\right)$ expresses the User's interest through an interval $I_F = \left[i_F^-, i_F^+\right)$.

For F to match n, some value contained in I_E must be also contained in I_F; I_F and I_E must *intersect*. Hence, the matching relation is based on a specific relationship of the involved intervals I_F and I_E. The same applies to the equality, intersection, disjointness, and cover relationships among two filters, which must be determined for notification routing optimizations.

A formal definition of interval relationships has been described in J. F. Allen's interval algebra. The following Subsection 4.1.1 will briefly introduce it, to then define interval filter relations and operations in Subsection 4.1.2. The conjunction of simple interval filters to multidimensional interval filters is then discussed in Subsection 4.1.3.

4.1.1 Formal Foundation: Interval Algebra

James F. Allen introduced in 1983 a *temporal interval algebra* [11], which has become a fundamental building block in temporal reasoning [125, 18], and has been found to be generally applicable to arbitrary types of intervals [126]. It is based on the observation, that only a limited number of topological relationships between two intervals

$A = [a^-, a^+], B = [b^-, b^+]$ are possible. By enumerating all possible orders of the begin and end points of the intervals, 13 mutually exclusive interval relationships can be identified (Figure 4.1). Allen first coined terms for the relationships, which are known today as *Allen's operators*.[1] Any given two intervals relate in exactly one of these ways. The relationships are read, e.g., "*A meets B*". They are often implemented as boolean-valued functions, e.g. $meets : \mathbb{I}^2 \rightarrow \{true, false\}$, where $\mathbb{I}$ denotes the set of all intervals.

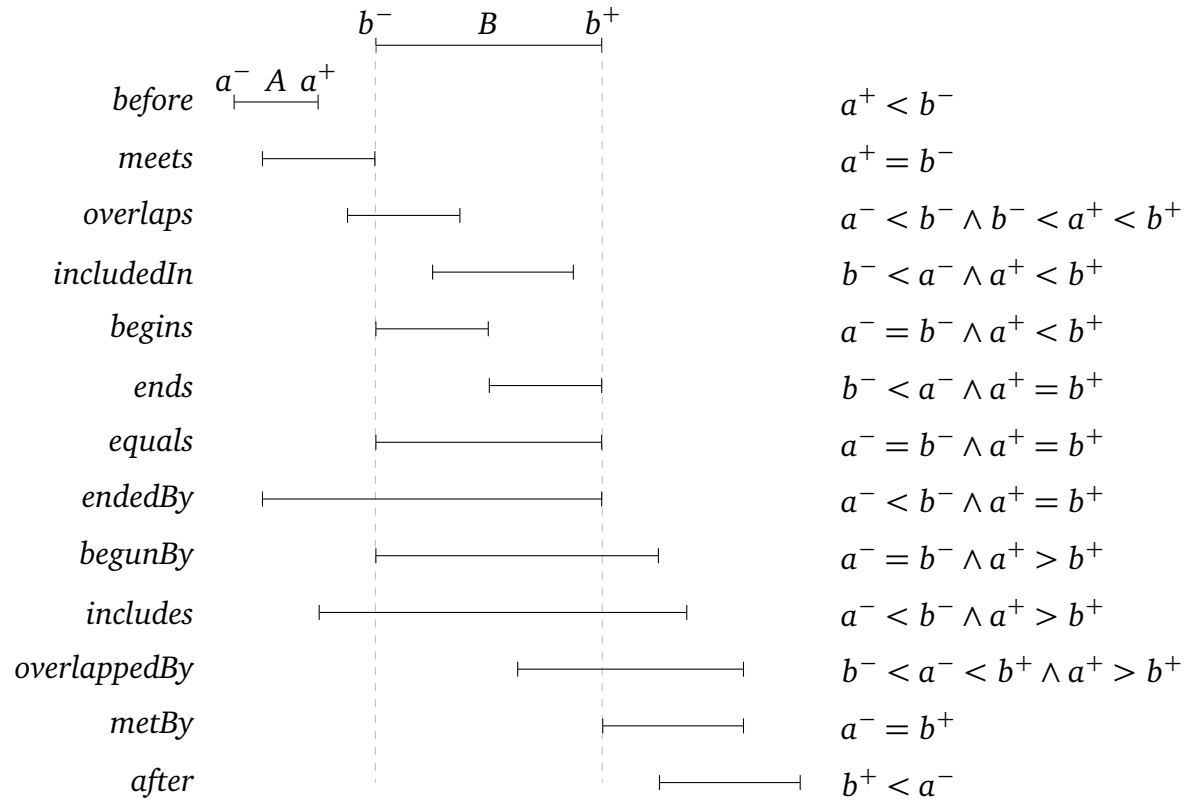


Figure 4.1.: Allen's operators: The 13 possible relationships between two intervals A, B. The 6 relationships below *equals* are the inverse of the ones above.

Using left-open or right-open intervals instead of closed ones, while not affecting the validity of Allen's interval algebra, changes the intuitive interpretation of *A meets B* (and of its inverse). In the case of closed intervals, A and B share the point $a^+ = b^-$. In the case of left-open intervals, this point is included in A only and in right-open intervals it is included in B.

For convenience, some of Allen's operators can be referred to together with its inverse with expressions like "*A* and *B meet*" meaning "*A meets B* or *A metBy B*". To facilitate further discussions, it also makes sense to additionally define some conve-

[1] Allen originally used the expressions *during*, *starts*, *finishes* instead of *includes*, *begins*, *ends*, when he introduced the operators exclusively focusing on temporal intervals.

nience named relations based on disjunctions of the 13 fundamental relationships. Let $A = [a^-, a^+), B = [b^-, b^+)$.

Definition Interval Disjointness: *Two intervals A, B are* disjoint *iff A before B or A after B or A and B meet.*

$$\text{disjoint}(A, B) = \text{before}(A, B) \vee \text{after}(A, B) \vee \text{meets}(A, B) \vee \text{metBy}(A, B)$$

Definition Interval Intersection: *Two intervals A, B* intersect *iff they are* not disjoint*, i.e. A and B overlap or A and B are equal or A includes/includedIn B or A begins/begunBy B or A ends/endedBy B.*

$$\text{intersect}(A, B) = \neg\text{disjoint}(A, B)$$

For left-open (or right-open) intervals A, B as we assume here, this is equivalent to

$$\text{intersect}(A, B) = a^- < b^+ \wedge b^- < a^+ \tag{4.1}$$

4.1.2 Interval Filter Relations

Notification matching of an interval filter $F = (I_F)$ and an interval notification $n = (I_E)$ as well as filter relationships between two interval filters $F = (I_F), G = (I_G)$ are defined in the following based on interval relationships.

Intervals of interval filters are generally assumed to be right-open, e.g. $I_F = [f^-, f^+)$. Hence, I_F is a set of consecutive discrete points $I_F = \{f^-, f^- + 1, \ldots, f^+ - 1\}$. It is required that $f^- < f^+$ such that I_F includes at least the point f^-. Notifications n for some interval event E are defined accordingly, $n = (I_E), I_E = [e^-, e^+)$.

Definition Interval Notification Matching: *An interval filter $F = (I_F)$ matches the notification n for an interval event $n = (I_E)$ iff I_F and I_E* intersect*.*

$$F(n) \Leftrightarrow \text{intersect}(I_F, I_E)$$

Definition Interval Filter Equality: *Two interval filters $F = (I_F), G = (I_G)$ are equal iff I_F and I_G are equal.*

$$F \equiv G \Leftrightarrow \text{equal}\left(I_F, I_G\right)$$

Definition Interval Filter Intersection: *Two interval filters* $F = \left(I_F\right), G = \left(I_G\right)$ *are intersecting iff* I_F *and* I_G *intersect.*

$$F \sqcap G \Leftrightarrow \text{intersect}\left(I_F, I_G\right)$$

Definition Interval Filter Disjointness: *Two interval filters* $F = \left(I_F\right), G = \left(I_G\right)$ *are disjoint iff* I_F *and* I_G *are disjoint.*

$$F \not\sqcap G \Leftrightarrow \text{disjoint}\left(I_F, I_G\right)$$

Definition Interval Filter Cover: *An interval filter* $F = \left(I_F\right)$ covers *an interval filter* $G = \left(I_G\right)$ *iff* I_F *and* I_G *are equal or* I_F *includes* I_G *or* I_G *begins* I_F *or* I_G *ends* I_F. *F properly covers G iff F covers G and* I_F *and* I_G *are not equal.*

$$F \sqsupseteq G \Leftrightarrow \text{equal}\left(I_F, I_G\right) \vee \text{includes}\left(I_F, I_G\right) \vee \text{begins}\left(I_G, I_F\right) \vee \text{ends}\left(I_G, I_F\right)$$
$$F \sqsupset G \Leftrightarrow \text{includes}\left(I_F, I_G\right) \vee \text{begins}\left(I_G, I_F\right) \vee \text{ends}\left(I_G, I_F\right)$$

4.1.3 N-Interval Filters

Assume we had notifications that expose the event's content as multiple intervals in distinct dimensions. This could for instance be notifications for stock ticker events, which abstract the event information to the price range of a specific stock and the temporal interval in which this price is/was/will be valid.[2] A notificiation for such a *multidimensional interval event* would take the form $n = \left(I_{E,i}\right)_{i=1}^{N}$ with intervals $I_{E,i}$ for each dimension i of the N exposed event characteristics. Subscription filters for those events take the form $F = \left(I_{F,i}\right)_{i=1}^{N}$. They shall in the following be referred to as N-interval filters, whereas simple interval filters as discussed before will be referred to as 1-interval filters, when the distinction is necessary. An N-interval filter can be regarded as a conjunction of N 1-interval filters $F_i = \left(I_{F,i}\right)$.

The matching of an N-interval filter and an N-interval notification and relationships between N-interval filters are defined as follows.

[2] Any other value modeled in the form of an interval is imaginable such as temperature or air pressure or some other measurement's range.

Definition N-Interval Notification Matching: *An N-interval filter $F = \left(I_{F,i}\right)_{i=1}^{N}$ matches the N-interval notification $n = \left(I_{E,i}\right)_{i=1}^{N}$, iff $I_{F,i}$ and $I_{E,i}$ intersect in each dimension i.*

$$F(n) \Leftrightarrow \bigwedge_{i=1}^{N} \text{intersect}\left(I_{F,i}, I_{E,i}\right)$$

Definition N-Interval Filter Equality: *Two N-interval filters $F = \left(I_{F,i}\right)_{i=1}^{N}, G = \left(I_{G,i}\right)_{i=1}^{N}$ are* equal *iff $I_{F,i}$ and $I_{G,i}$ are equal in each dimension i.*

$$F \equiv G \Leftrightarrow \bigwedge_{i=1}^{N} \text{equal}\left(I_{F,i}, I_{G,i}\right)$$

Definition N-Interval Filter Intersection: *Two N-interval filters $F = \left(I_{F,i}\right)_{i=1}^{N}, G = \left(I_{G,i}\right)_{i=1}^{N}$ are* intersecting *iff $I_{F,i}$ and $I_{G,i}$ intersect in each dimension i.*

$$F \sqcap G \Leftrightarrow \bigwedge_{i=1}^{N} \text{intersect}\left(I_{F,i}, I_{G,i}\right)$$

Definition N-Interval Filter Disjointness: *Two interval filters $F = \left(I_{F,i}\right)_{i=1}^{N}, G = \left(I_{G,i}\right)_{i=1}^{N}$ are* disjoint *iff $I_{F,i}$ and $I_{G,i}$ are disjoint in at least one dimension i.*

$$F \not\sqcap G \Leftrightarrow \bigvee_{i=1}^{N} \text{disjoint}\left(I_{F,i}, I_{G,i}\right)$$

Definition N-Interval Filter Cover: *An N-interval filter $F = \left(I_{F,i}\right)_{i=1}^{N}$ covers an N-interval filter $G = \left(I_{G,i}\right)_{i=1}^{N}$ iff $I_{F,i}$ and $I_{G,i}$ are equal or $I_{F,i}$ includes $I_{G,i}$ or $I_{G,i}$ begins $I_{F,i}$ or $I_{G,i}$ ends $I_{F,i}$ in each dimension i. F* properly covers *G if F covers G and $I_{F,i}$ and $I_{G,i}$ are not equal in at least one dimension i.*

$$F \sqsupseteq G \Leftrightarrow \bigwedge_{i=1}^{N} \text{equal}\left(I_{F,i}, I_{G,i}\right) \vee \text{includes}\left(I_{F,i}, I_{G,i}\right) \vee \text{begins}\left(I_{G,i}, I_{F,i}\right) \vee \text{ends}\left(I_{G,i}, I_{F,i}\right)$$

$$F \sqsupset G \Leftrightarrow F \sqsupseteq G \wedge \exists i \left(\neg \text{equal}\left(I_{F,i}, I_{G,i}\right)\right)$$

4.2 Spatial Filters

Notifications for spatial events implement some characteristic of the event as a 2-dimensional spatial region. In our case, this is the horizontal spatial effectivity of the event. Such spatial notifications are modeled for the purpose here by a polygon P_E, $n = (P_E)$. A subscription filter for such a spatial event is implemented by a *spatial filter F* with a filter polygon P_F representing the region of interest to the User, $F = (P_F)$. For this point of the discussion, the polygons are assumed to be in BREP and simple. Geospatial issues are discussed in Section 4.3.

Obviously, as with intervals, the conditions for notfication matching and filter relationships depend on the topological relationship of the regions represented by P_E and P_F. *Topological spatial relations* are a well researched field in the area of (geo-)spatial data. From this field, we use *4-intersection logic* for the formalization of spatial filter operations and relations. It is briefly introduced in the following Subsection 4.2.1 before presenting spatial filter relations in Subsection 4.2.2 and discussing Computational Geometry algorithms required for spatial filter handling in Subsection 4.2.3.

4.2.1 Formal Foundation: Topological Spatial Relations

Topology is a branch of geometry that is concerned with geometric properties that remain invariant under certain operations like rotation, translation and scaling. It describes spatiality through relationships between topological elements. Formal frameworks for the distinction of toplogical relationships between spatial objects are a widely studied field. M. Egenhofer, E. Clementini, P. Di Felice, R. Franzosa *et al.* have all considerably contributed to this field (see, e.g., references [50, 51, 53, 52, 54, 39]).

Point set topology supplies a formal framework to distinguish the *boundary*, *interior*, and *exterior* of a spatial object P, denoted with ∂P, P°, and P^- respectively [53, 52]. With 4-intersection logic [52], topological relationships between two regions P, Q are distinguished by whether or not their boundaries and interiors intersect, resulting in four pairwise intersection tests, commonly represented in a 2x2 *intersection matrix* (4IM).

$$4\text{IM}(P,Q) = \begin{pmatrix} \partial P \cap \partial Q & P^\circ \cap \partial Q \\ \partial P \cap Q^\circ & P^\circ \cap Q^\circ \end{pmatrix}$$

By considering the topological invariants empty ($\emptyset$) and non-empty ($\neg\emptyset$), 2^4 combinations can be distinguished, eight of which can be realized for two simple regions in

the plane [51] (Figure 4.2). The relations are read, e.g., "P covers Q". They are often implemented as boolean-valued functions, e.g., $covers : \mathbb{P}^2 \rightarrow \{true, false\}$, where $\mathbb{P}$ denotes the set of all regions.

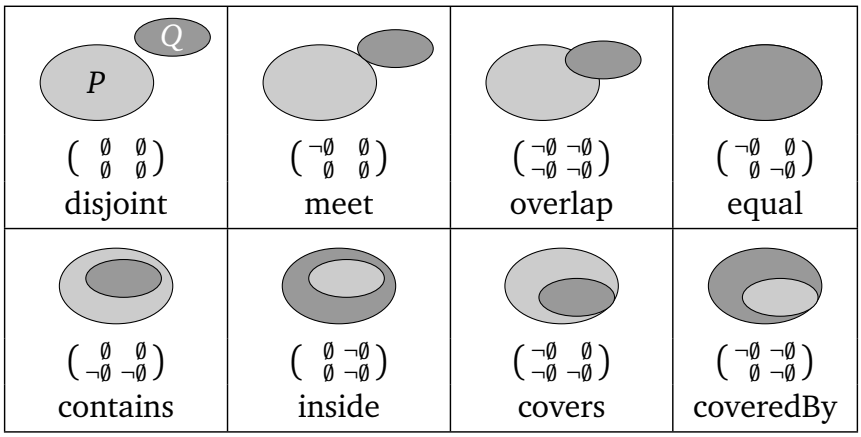


Figure 4.2.: The eight topological spatial relations between two simple regions in the plane with natural language names.

As for intervals, we also define a convenience named relation "intersect" as a subjunction of basic relationships.

Definition Region Intersection: *Two regions P,Q* intersect *iff P and Q are equal or P and Q overlap or P contains/inside Q or P covers/coveredBy Q.*

$$\text{intersect}(P,Q) = \text{overlap}(P,Q) \vee \text{equal}(P,Q) \vee \text{contains}(P,Q) \vee \text{inside}(P,Q) \vee \text{convers}(P,Q) \vee \text{coveredBy}(P,Q$$

Note that, as an important difference from the definition for intervals, neither region intersection nor disjointness includes the *meet* relationship. This is because the named relationship *disjoint* is a basic relationship for regions whereas it is for intervals a convenience relationship defined as subjunction of other basic relationships. For regions, *disjoint* and *meet* and *intersect* are mutually exclusive and complete as disjunction.

4.2.2 Spatial Filter Relations

Spatial filter operations and relationships can now be discussed based on the spatial topological relations formally introduced in the previous section. For the following discussion, let $F = (P_F), G = (P_G)$ be spatial filters and $n = (P_E)$ a notification for a spatial event E (hereafter referred to as *spatial notification*). The simple polygons P_F, P_G, P_E represent the regions of the filters and the notification.

Definition Spatial Notification Matching: *A spatial filter* $F = (P_F)$ *matches a spatial notification* $n = (P_E)$ *iff* P_F *and* P_E intersect.

$$F(n) \Leftrightarrow \text{intersect}(P_F, P_E)$$

Definition Spatial Filter Equality: *Two spatial filters* $F = (P_F), G = (P_G)$ *are equal iff* P_F *and* P_G *are equal.*

$$F \equiv G \Leftrightarrow \text{equal}(P_F, P_G)$$

Definition Spatial Filter Intersection: *Two spatial filters* $F = (P_F), G = (P_G)$ *are intersecting iff* P_F *and* P_G *intersect.*

$$F \sqcap G \Leftrightarrow \text{intersect}(P_F, P_G)$$

Definition Spatial Filter Disjointness: *Two spatial filters* $F = (P_F), G = (P_G)$ *are disjoint iff* P_F *and* P_G *are disjoint or meet.*

$$F \not\sqcap G \Leftrightarrow \text{disjoint}(P_F, P_G) \vee \text{meet}(P_F, P_G)$$

Definition Spatial Filter Cover: *A spatial filter* $F = (P_F)$ *covers a spatial filter* $G = (P_G)$ *iff* P_F *and* P_G *are equal or* P_F *contains* P_G *or* P_F *covers* P_G. *F properly covers G iff* P_F *contains* P_G *or* P_F *covers* P_G.

$$F \sqsupseteq G \Leftrightarrow \text{equal}(P_F, P_G) \vee \text{contains}(P_F, P_G) \vee \text{covers}(P_F, P_G)$$
$$F \sqsupset G \Leftrightarrow \text{contains}(P_F, P_G) \vee \text{covers}(P_F, P_G)$$

4.2.3 Computational Geometry Algorithms

Whereas the implementation of interval filter operations is straight-forward, the implementation of the filter methods and relation tests for spatial filters and notifications requires algorithms from the field of Computational Geometry (CG). In the following, these shall be presented briefly. A specific focus is on their computational complexity because the algorithms' run time widely depends on the characteristics of the involved geometries, which is not the case with interval filters, where all operations on N-interval filters obviously require $O(N)$ time.

If, in the following, uppercase Q denotes a polygon, let lowercase q denote its *complexity*, i.e., the number of vertices and line segments the polygon boundary is defined by.

We first introduce *Point in Polygon* and *Segment Intersection* as generic basic CG problems, before discussing the algorithms required in the filter operations in our application that partly build upon these basic problems.

Point in Polygon

One of the most basic questions in Computational Geometry is whether a point P is inside a polygon Q. The problem is such prevalent that it is widely known as the *point-in-polygon problem (PIP)*. Proposals for algorithms trace back to the beginnings of computer graphics [162]. Today, different most efficient algorithms are known depending on the polygon's properties [145]. The *ray casting algorithm* (or *even-odd rule*) is the most efficient one for simple, not necessarily convex, polygons. It takes $O(q)$ time.

The idea is based on the *Jordan Curve Theorem* [175], which essentially says that if a test ray from P to infinity (in any direction) crosses Q's boundary an odd number of times, P is inside Q. It is outside if the number of edge crossings is even. Figure 4.3 shows examples: P_1 is outside Q and the ray from it crosses Q's boundary four times, whereas P_2 is inside Q and the ray crosses three edge segments of Q.

When implementing the algorithm, special attention has to be paid for pathological cases, e.g. if the ray crosses a boundary vertex or overlaps with an edge (as in the case of P_3). It has been shown that simple conventions (e.g. considering all vertices on the ray to be above the ray) can be applied in these cases [90].

As a final note, Preparata and Shamos showed in their 1985 CG text book [150] that $\Omega(q)$ and $\Omega(q)$ are also the lower time bounds for PIP algorithms for general, simple polygons, because in any case, every edge of Q has to be visited.

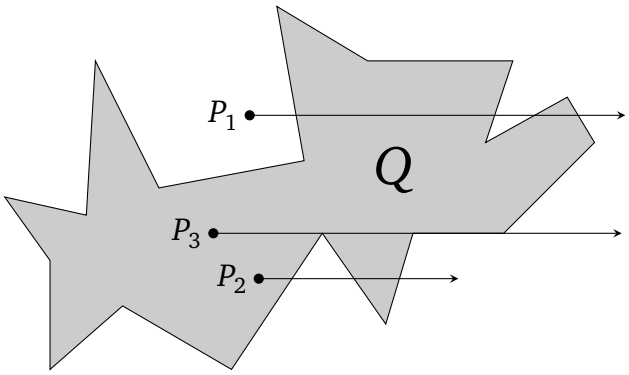


Figure 4.3.: Point-in-polygon problem: Ray casting algorithm for a concave polygon

Segment Intersection Test

Another traditional class of problems in Computational Geometry is geometric intersection [132], i.e., the problems of either finding all intersections between two geometries or detecting an intersection. It has a number of applications, for instance to decide whether a given linestring forms the boundary of a simple polygon. This question reduces to detecting whether the linestring is self-intersecting, i.e., whether any two segments intersect.

For our purpose, the specific problem is, given two simple polygons P,Q, to decide whether a boundary segment of P intersects with a boundary segment of Q. This is needed in the matching test of spatial filters and notifications described in the next subsection.

B. Chazelle presented in [35] a linear-time algorithm for the triangulation of a simple polygon R. Since the algorithm crashes when the polygon is not simple, it can be used to detect segment intersections in a linestring ℓs by assuming it to be the boundary of the polygon R and running the triangulation algorithm. This approach can in turn be used for our problem. By joining P and Q into a single closed chain by a narrow channel (Figure 4.4), the polygon R is created. If P and Q are simple, R is also simple if the boundaries of P and Q do not contain mutually intersecting segments. Running Chazelle's triangulation algorithm on R will thus detect intersections of the polygon boundaries by crashing [132].

The quite complex triangulation algorithm is not presented here for brevity. However, the conclusion at this point is that using B. Chazelle's algorithm the boundary segment intersection test requires $O\left(p+q\right)$ time in the worst case.

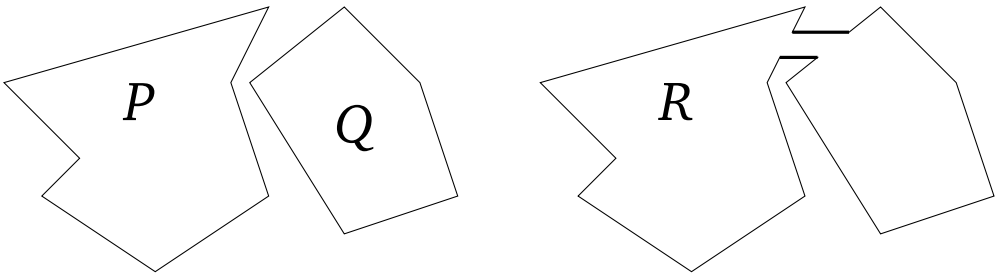


Figure 4.4.: Joining two simple polygons P,Q into a polygon R by a narrow channel to determine whether the boundaries of P and Q intersect by determining whether R is simple.

Polygon Intersection Test

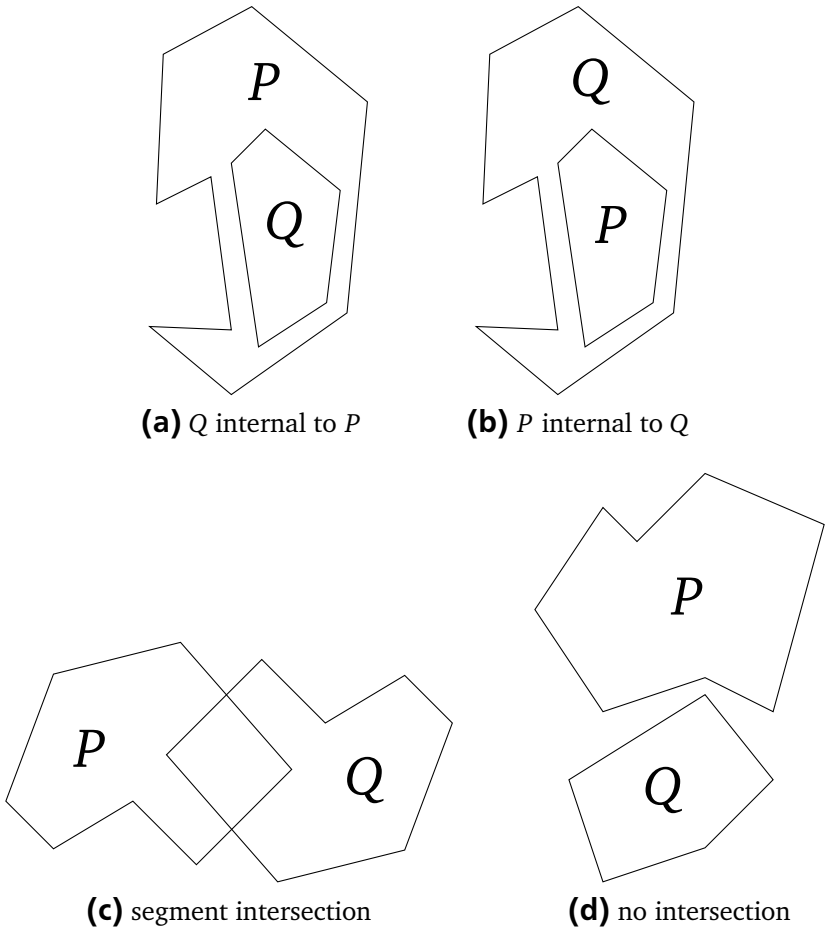
The matching test of a spatial filter $F = (P_F)$ and a notification $n = (P_E)$ requires determining whether the regions represented by P_F and P_E intersect. This *polygon intersection test* is most efficiently realized based on PIP and segment intersection tests [150] (Figure 4.5). We see that P and Q intersect iff every vertex of Q is internal to P (a) or vice versa (b) or some edge of P intersects an edge of Q (c).

Hence, the polygon intersection test for two polygons P,Q requires two point-in-polygon tests and, if these fail, a segment intersection test. It takes therefore $O(p) + O(q) + O(p+q)$ in the worst case.

Topological Spatial Relation Test

The common way to implement topology tests and operations on geometries is by means of an auxiliary data structure: the Doubly-Connected Edge List (DCEL). It was first proposed by Preparata and Shamos [150] and is used to describe the topological structure of a geometry. It contains a record of each vertex, edge, and face. Each vertex stores its coordinates and the edges which end in the vertex. Each edge stores two vertices, the right and left faces and usually the previous and following edges. Examples of records within such an edge list can be found in reference [47]. Let $N = p + q$. Building the DCEL for two polygons P,Q takes $O(N \log N + k \log N)$ time, where k is the number of intersections of the boundaries of P and Q [108].

The intersection matrix 4IM(P,Q) can simply be derived from the DCEL. Hence, the time required for building the DCEL is also the time required for determining the topological relation between two regions represented by polygons in BREP.



(a) Q internal to P **(b)** P internal to Q

(c) segment intersection **(d)** no intersection

Figure 4.5.: Polygon intersection

Envelopes

The most common approach to gain efficiency in geometric operations is through the exploitation of *envelopes*. The envelope operation projects a polygon P onto the axes of the coordinate system resulting in two intervals, $envelope : P \rightarrow (I_X, I_Y), X = [x_{min}, x_{max}], Y = [y_{min}, y_{max}]$. The result is commonly represented as the minimal rectangle enclosing P (Figure 4.6), and $envelope(P)$ is referred to as P's *bounding box* or *envelope*.[3]

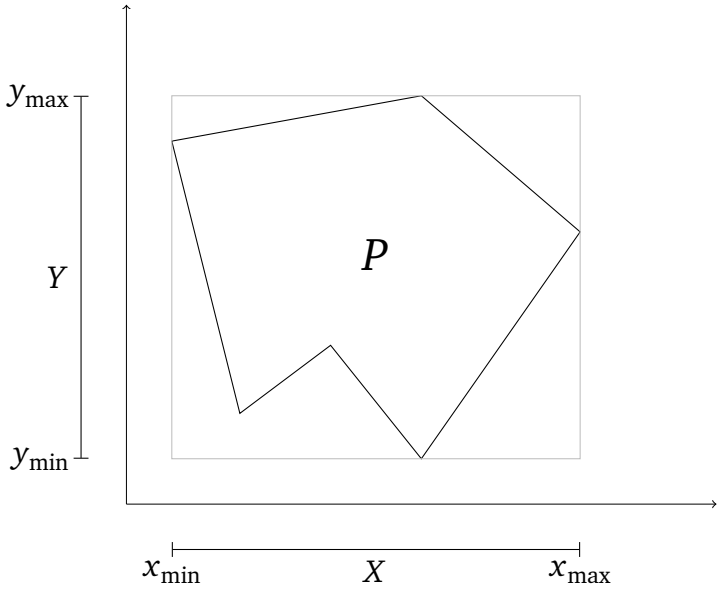


Figure 4.6.: Envelope of a polygon P

The bounding box is a simple approximation of the actual geometry that can be used for intersection and inclusion candidate selection before applying any of the above algorithms on the actual geometry. It can be decided in constant time whether a point is inside the envelope or whether two envelopes intersect. Only if these tests succeed, the PIP test, segment intersection, or topological relation tests on the actual geometry must be performed. The envelope operation itself obviously runs in $O(p)$ (and $\Omega(p)$) time, because every vertex has to be visited to find the overall minimal x and y values. In practice, it is typically performed only once and the envelope is stored with P in constant space to be reused for all future operations on P.

Regarding our filter types, the envelope operation can be used to simplify a spatial filter $F = (P_F)$ to a 2-interval filter F' by approximating the polygon P_F with its envelope,

[3] The term *Minimum Bounding Rectangle (MBR)* is also often used instead of bounding box or envelope. It is avoided here, because in other spaces than 2-dimensional Euclidean space, the bounding box is often not a rectangle. This applies most notably to the surface of the Earth as discussed in the next section.

$F' = envelope(P_F) = (I_{F,1}, I_{F,2})$. To remain consistent with our implementation of intervals, the envelope operation can be implemented such that it results in two right-open intervals over spatial point types X, Y, $I_X = [x^-, x^+), I_Y = [y^-, y^+)$. This is used in the approach to imperfect merging of spatial filters presented later in Section 5.1.3.

4.3 Geospatial Filters

Thus far, it was left open how the filter or notification polygons were implemented, i.e., whether an implementation in $\mathbb{R}^2$ is assumed or in a geographic coordinate system. In fact, the definition of filter relations and the matching operation is solely based on topology and therefore independent of the specific implementation of geometries. The usual CG algorithms however assume $\mathbb{R}^2$ space. Geometry vertices are specified with coordinates in a Cartesian plane, and edges are linear interpolations between these vertices. In their typical implementation these algorithms can fail if the edge interpolation is not linear.

The algorithms required for the filter operations have to be adapted to correctly handle geometries specified in geographic coordinates, and line interpolations other than linear. The required adaptation of the algorithms is presented from Subsection 4.3.2 on after deriving equations for the different path types in the following subsection.

4.3.1 Aerial Navigation Path Interpolation

A spherical Earth model is assumed in the following. The radius R of the Earth is canceled out in all following equations, hence the unit sphere is assumed without loss of generality.

A linear interpolation between the points $P = (\phi_P, \lambda_P)$ and $Q = (\phi_Q, \lambda_Q)$ is given by the familiar linear equation:

$$\phi = \phi_P + \frac{\lambda - \lambda_P}{\lambda_Q - \lambda_P} \left(\phi_Q - \phi_P\right) \tag{4.2}$$

This path type is never used in aerial navigation. Its practical relevance is only for the approximation of orthodromes and loxodromes as discussed later in Subsection 4.3.5.

Loxodrome

The Mercator projection equation and a loxodrome function are closely related. In a Mercator projection, meridians are projected to parallel lines as in the Plate Caree. But moreover, distances are stretched by the reciprocal factor of shrinkage of the parallels

toward the poles. On the Earth, the size of the parallels (radius and circumference) shrinks from the equator toward the poles at a factor of $\cos(\phi)$. In the Mercator projection, this shrinking is canceled through stretching distances by a factor of the reciprocal of the cosinus function, the secant function $\sec(\phi)$. This results in equally spaced, parallel meridians, and in parallels, which are not equally spaced anymore, but are placed with growing distance toward the poles (see Figure 3.16b[4]).

Since loxodromes are straight lines in the Mercator projection, a loxodrome function can be approached by looking at the Mercator projection's properties. While the local stretching factor at a point at latitude ϕ is $\sec(\phi)$, the total stretching of a distance in direction north-south is the integral of the local stretching factor,

$$\Sigma(\phi) = \int_0^{\phi} \sec(\phi) d\phi = \ln\left(\left|\sec\phi + \tan\phi\right|\right) = \ln\left(\tan\frac{1}{2}\left(\frac{\pi}{2} + \phi\right)\right) \tag{4.3}$$

so that the ϕ-parallel is placed at north-south distance $\Sigma(\phi)$ from the equator. In a Mercator projection, map coordinates are given by $x_{\text{map}} = \lambda, y_{\text{map}} = \Sigma(\phi)$. A straight line through $P = (\phi_P, \lambda_P)$ thus takes the form

$$\Sigma = \Sigma(\phi_P) + m(\lambda - \lambda_P). \tag{4.4}$$

Given another point $Q = (\phi_Q, \lambda_Q)$ on this loxodrome, the slope m is given by

$$m = \frac{\Sigma(\phi_Q) - \Sigma(\phi_P)}{\lambda_Q - \lambda_P},$$

and the loxodrome's bearing Φ is given by

$$\Phi = \operatorname{arccot}(m).$$

The equation of the loxodrome through P is found by inverting Equation (4.3) and inserting Equation (4.4),

$$\begin{aligned} \phi &= 2\arctan\left(e^{\Sigma(\phi)}\right) - \frac{\pi}{2} \\ &= \frac{\pi}{2} - 2\arctan\left(e^{-\Sigma(\phi)}\right) \\ &= \frac{\pi}{2} - 2\arctan\left(e^{-\left(\Sigma(\phi_P) + m(\lambda - \lambda_P)\right)}\right) \end{aligned} \tag{4.5}$$

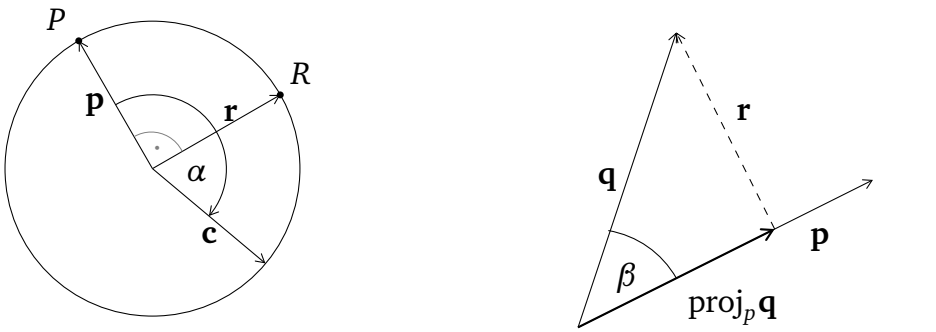
[4] Section 3.3.2, Page 59

There are an infinite number of loxodromes between any two points where $\phi_P \neq \phi_Q$, differing in the number of times they encircle the Earth. Assuming the *shortest* loxodrome is to be found, special attention is to be paid to pathological cases (e.g., when it crosses the International Date Line) [10]. In the case where $\phi_P = \phi_Q$, the bearing $\Phi = 0$, and the loxodrome equals a line of constant latitude.

Orthodrome

Given two perpendicular vectors $\mathbf{p}, \mathbf{r}$ of points P, R on the great circle, great circle points $\mathbf{c}$ can be expressed as a function of the angle α between $\mathbf{p}$ and $\mathbf{c}$ toward $\mathbf{r}$ (Figure 4.7a):

$$\mathbf{c}(\alpha) = \mathbf{p}\cos\alpha + \mathbf{r}\sin\alpha \tag{4.6}$$



(a) Spanning vectors $\mathbf{p}, \mathbf{r}$ **(b)** Construction of $\mathbf{r}$ from $\mathbf{p}$ and $\mathbf{q}$

Figure 4.7.: Geometric construction of a great circle

If the great circle function is to be found given two arbitrary great circle points P, Q, $\mathbf{p}$ and $\mathbf{q}$ are not necessarily perpendicular. Therefore, we must find a point R such that $\mathbf{r}$ is perpendicular to $\mathbf{p}$ in the direction of $\mathbf{q}$ (Figure 4.7b). We first find the angle β between $\mathbf{p}$ and $\mathbf{q}$ with the dot product, $\beta = \arccos(\mathbf{p} \cdot \mathbf{q})$ (because of $|\mathbf{p}| = |\mathbf{q}| = 1$), to then project $\mathbf{q}$ onto $\mathbf{p}$:

$$\text{proj}_\mathbf{p}\mathbf{q} = \frac{\mathbf{p}\cdot\mathbf{q}}{\mathbf{p}\cdot\mathbf{p}}\mathbf{p} = \frac{\cos\beta}{\cos 0}\mathbf{p}$$

The resulting vector is subtracted from $\mathbf{q}$:

$$\mathbf{r} = \mathbf{q} - \text{proj}_\mathbf{p}\mathbf{q} = \mathbf{q} - \mathbf{p}\cos\beta \tag{4.7}$$

Finally, $\mathbf{r}$ is normalized by dividing by its length, $|\mathbf{r}| = \sin\beta$ (because of $\sin\beta = \frac{|\mathbf{r}|}{|\mathbf{q}|}$). Inserting in Equation (4.6) gives

$$\mathbf{c}(\alpha) = \mathbf{p}\cos\alpha + \frac{\mathbf{q} - \mathbf{p}\cos\beta}{\sin\beta}\sin\alpha \tag{4.8}$$

Obviously, $\mathbf{c}(0) = \mathbf{p}$ and $\mathbf{c}(\beta) = \mathbf{q}$. The points on the orthodrome between P and Q are found at $\mathbf{c}(0 \leq \alpha \leq \beta)$.

An equation for the latitudes of the great circle points as a function of longitude can be found in Ed Williams' *Aviation Formulary* [178]. It is quoted here for completeness:

$$\phi = \arctan\left(\frac{\sin\phi_P\cos\phi_Q\sin(\lambda - \lambda_Q) - \sin\phi_Q\cos\phi_P\sin(\lambda - \lambda_P)}{\cos\phi_P\cos\phi_Q\sin(\lambda_P - \lambda_Q)}\right) \tag{4.9}$$

4.3.2 Envelopes of Paths

The envelope operation is expected to return in two intervals the minimum and maximum extent values of the geometry. In the case of geographic coordinates, these are the intervals $I_\Lambda = \left[\lambda^-, \lambda^+\right], I_\Phi = \left[\phi^-, \phi^+\right]$. The common implementation of *envelope*(P) simply inspects all boundary vertices of P to find the minimal and maximal values. In the case of orthodrome interpolation of the edges, the maximum point of the geometry may however be located on the edge. Figure 4.8 shows the linear interpolation, loxodrome, and orthodrome between New York, Moscow, and Dakar in a Plate Carée, which treats (ϕ, λ) as Cartesian coordinates. Obviously, the maximum latitude of the triangle is not found at either vertex but on a boundary segment, if orthodromes are used as boundary segments.

The envelope operation has to be adapted to account for these cases. Since different interpolations may be specified for each line in the boundary linestring, each line must be checked individually.

Hence, the task is to find the extreme latitude and longitude values $\phi^-, \phi^+, \lambda^-, \lambda^+$ of a line ℓ depending on its interpolation type. Let $\ell = (P,Q), P = (\phi_P, \lambda_P), Q = (\phi_Q, \lambda_Q)$. We assume $\lambda_P < \lambda_Q$ without loss of generality. Considering the path functions in Equations (4.2), (4.5), and (4.9), we see that λ^-, λ^+ are found at P,Q in every case: All equations are formulas for latitude as a function of longitude, $\phi(\lambda)$, and points on the line are found for $\lambda_P \leq \lambda \leq \lambda_Q$. Hence,

$$\begin{aligned} \lambda^- &= \lambda_P \\ \lambda^+ &= \lambda_Q \end{aligned} \tag{4.10}$$

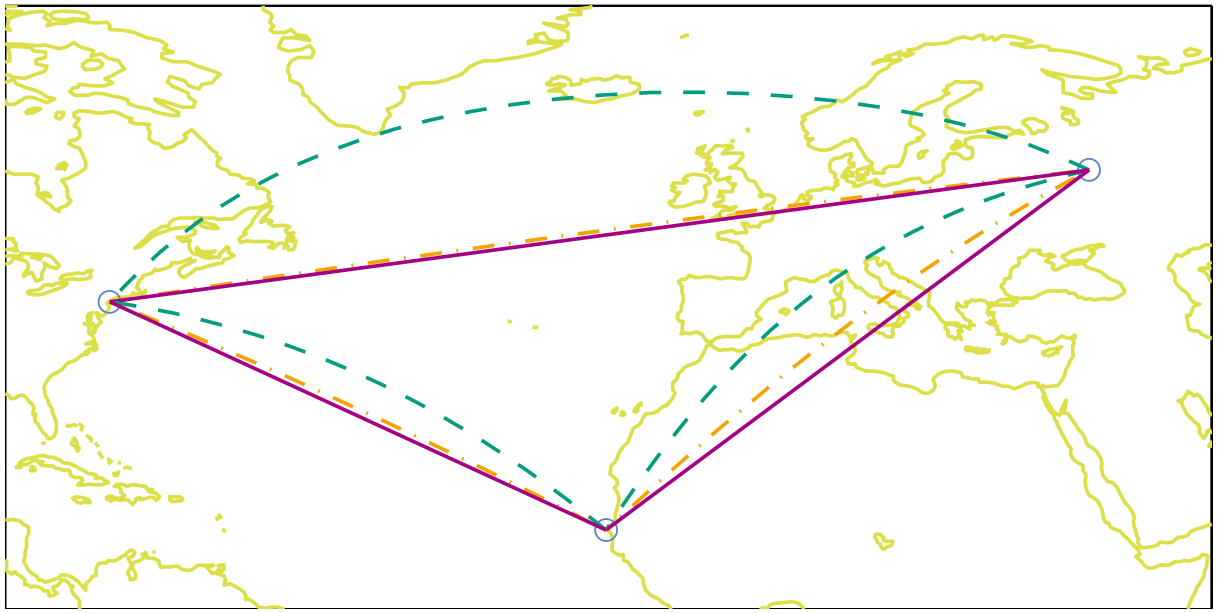


Figure 4.8.: Triangle New York City – Moscow – Dakar with different line interpolations: linear (solid), loxodrome (dash-dotted), and orthodrome (dashed) in a Plate Carée projection

for all path types.

For linear interpolations, the extreme latitude values are of course also found at P, Q. This is also the case for the loxodrome interpolation: Equation (4.5) merely consists of an inverse tangent, which is a monotonic function. For a linearly interpolated or loxodrome line, the extreme latitude values of ℓ are therefore

$$\begin{aligned} \phi^- &= \min(\phi_P, \phi_Q) \\ \phi^+ &= \max(\phi_P, \phi_Q). \end{aligned} \tag{4.11}$$

Thus remain only the extreme latitude values of the orthodrome interpolation. Orthodromes are always "arched" to the north on the northern hemisphere and to the south on the southern hemisphere. Therefore, $\phi^- = \min(\phi_P, \phi_Q)$ if $\phi_P \geq 0$ and $\phi_Q \geq 0$, and $\phi^+ = \max(\phi_P, \phi_Q)$ if $\phi_P \leq 0$ and $\phi_Q \leq 0$. In the remaining cases, the situation is more complicated.

We approach a formula by considering the Cartesian coordinate's z values of great circle points. The equation for z values of great circle points is found by combining Equation (3.2) and Equation (4.6). Its derivatives are:

$$
\begin{aligned}
z(\alpha) &= \quad z_p \cos\alpha + z_r \sin\alpha \\
z'(\alpha) &= -z_p \sin\alpha + z_r \cos\alpha \\
z''(\alpha) &= -z_p \cos\alpha - z_r \sin\alpha
\end{aligned}
$$

The northernmost and southernmost point of the great circle are those with extreme z. Extrema of $z(\alpha)$ are at α_E if $z'(\alpha_E) = 0$. Hence,

$$
\begin{aligned}
& z_P \sin\alpha_E = z_R \cos\alpha_E \\
\Leftrightarrow \quad & \frac{\sin\alpha_E}{\cos\alpha_E} = \frac{z_R}{z_P} \\
\Leftrightarrow \quad & \alpha_E = \arctan\left(\frac{z_R}{z_P}\right)
\end{aligned}
\tag{4.12}
$$

The equator must be considered a pathological case where $z_P = z_R = 0$. Every other great circle has two antipodel extreme points E and E' at α_E and at the antipodal point $\alpha_{E'} = \alpha_E \pm 180°$ with $\phi_E = -\phi_{E'}$.

As an example, consider the orthodrome between New York City and Moscow from Figure 4.8. Let $P = (40.72°, -74°)$ (New York City) and $Q = (55.75°, 37.62°)$ (Moscow). Using Equations (3.1), (4.7), and (4.12), we find $\alpha_E = 43.822°$ and $\alpha_{E'} = -136.178°$. On the northern hemisphere, we are only interested in the maximum, i.e., the extremum where $z''(\alpha) < 0$. This is the case for α_E. Using Equation (4.8) and (3.2), we find the northernmost point of the orthodrome at $\lambda_E = -11.18°$ and $\phi_E = 62.05°$. Since $\lambda_P < \lambda_E < \lambda_Q$, the point lies on the orthodrome between New York City and Moscow. The maximum latitude is thus in this case $\phi^+ = \phi_E$.

Let $E = (\phi_E, \lambda_E)$ and $E' = (\phi_{E'}, \lambda_{E'})$ denote the northernmost and southernmost point, respectively, of the great circle through P and Q, calculated as above. The extreme latitudes of the orthodrome ℓ are then given by

$$
\begin{aligned}
\phi^- &= \begin{cases} \phi_{E'} & \text{if } \phi_P \le 0 \wedge \phi_Q \le 0 \wedge \lambda_P < \lambda_{E'} < \lambda_Q \\ \min\left(\phi_P, \phi_Q\right) & \text{else} \end{cases} \\
\phi^+ &= \begin{cases} \phi_E & \text{if } \phi_P \ge 0 \wedge \phi_Q \ge 0 \wedge \lambda_P < \lambda_E < \lambda_Q \\ \max\left(\phi_P, \phi_Q\right) & \text{else} \end{cases}
\end{aligned}
\tag{4.13}
$$

4.3.3 Geographic Point in Polygon

The ray casting algorithm to determine whether a point P is inside a polygon Q requires determining the number of intersections of Q's boundary and a virtual ray starting from P. This algorithm is adjusted to meet the requirements of geographical data as follows.

Let $P = (\phi_P, \lambda_P)$. Initially, it is checked whether P is inside *envelope*(Q). If it is not, P cannot be inside Q. Next, a virtual test ray from P due north (toward the North Pole) is constructed, and its intersections with Q's boundary are counted. Iterating over the edge segments $\ell_i = (q_i, q_{i+1})$, an envelope test is performed again, to avoid the costly trigonometric calculations. Let λ_i denote the longitude of the boundary vertex q_i. Only if $\lambda_i \leq \lambda_P \leq \lambda_{i+1}$ (assuming $\lambda_i \leq \lambda_{i+1}$ without loss of generality), the northerly test ray can possibly intersect the segment. Using ℓ_i's envelope that has been calculated in the course of calculating *envelope*(Q), it can also be checked if λ_P is within the envelope's latitude interval. In the positive case, the actual path interpolation equations are regarded. Using Equations (4.2), (4.5), and (4.9), let $\phi_{\ell_i}(\lambda_P)$ denote the latitude at which ℓ_i crosses the λ_P meridian. If P is in the northern hemisphere, the test ray intersects ℓ_i iff $\phi_{\ell_i}(\lambda_P) > \phi_P$. The opposite is true if it is in the southern hemisphere.

Again, special precaution has to be taken for pathological cases such as when Q includes the North Pole. In this case, a southerly test ray can be used, and the algorithm is adapted appropriately. A polygon including both poles should simply not be permitted. This limitation should not be a problem for almost every useful application.

4.3.4 Map Edge Singularity

A specific pathological case for the above equations is the "map edge singularity": Longitude is only defined in the range $[-180°, 180°]$. If some polygon edge segment $\ell = (P, Q)$ crosses the international date line at $\lambda = \pm 180°$ such that, e.g., $P = (\phi_P, 179°), Q = (\phi_Q, -179°)$, special attention has to be paid to avoid that the line is treated as $\ell = (Q, P)$, almost entirely encircling the Earth. This also plays an important role for the envelope operation, because an envelope calculated the usual way as above would result in a huge longitude interval, $I_\Lambda = [-179°, 179°]$ in this example.

A pragmatic solution for this problem is splitting the polygon along the $\pm 180°$ meridian such that two or more polygons[5] $P', P'', \ldots$ result. As can be seen in the example in Figure 4.9, at least one additional edge segment (K_1, K_2) for each of the resulting polygons has to be created. This requires calculating the intersection points K_i with the $\pm 180°$ meridian, for which Equations (4.2), (4.5), and (4.9) are used.

[5] More than two polygons result if P is concave and more than two edge segments of P cross the $\pm 180°$ meridian.

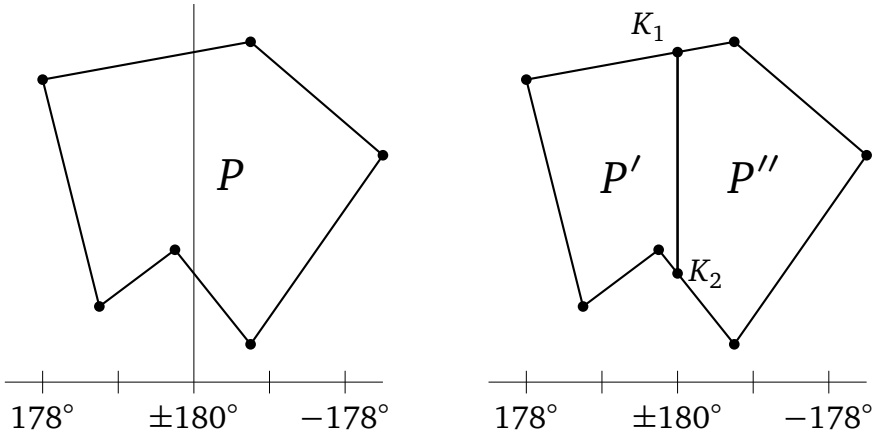


Figure 4.9.: Splitting a polygon along the ±180° meridian to avoid pathological cases

4.3.5 Path Approximation

A loxodrome is considerably longer than an orthodrome. Whereas the distance between two points P,Q on a great circle is given by [178]:

$$d_{\mathrm{O}} = \arccos\left(\sin\left(\phi_P\right)\sin\left(\phi_Q\right) + \cos\left(\phi_P\right)\cos\left(\phi_Q\right)\cos\left(\lambda_P - \lambda_Q\right)\right), \tag{4.14}$$

the loxodrome distance is simply the north-south distance $\left|\phi_P - \phi_Q\right|$ times the absolute value of the secant of the bearing Φ:

$$d_{\mathrm{L}} = \left|\sec\Phi\right| \cdot \left|\phi_P - \phi_Q\right|. \tag{4.15}$$

Both distance equations give a result in angle units. They can be expressed in distance units considering that one arc minute equals one nautical mile (nm).[6]

In our running example, the flight path between New York City and Moscow is thus either $d_{\mathrm{O}} \approx 4052\,\mathrm{nm}$ or $d_{\mathrm{L}} \approx 4506\,\mathrm{nm}$ long.

Since the loxodrome is however the only path type that can be navigated by conventional navigation means, longer routes are often navigated with approximated orthodromes: Loxodromes are flown between waypoints along the great circle track. Figure 4.10 shows the North Atlantic Tracks[7], where waypoints are defined at the intersection points of the orthodromes between start and end point and the $20°, 30°, 40°, 50°$

[6] A nautical mile is defined as a 21,600th of the Earth's circumference C, assuming a spherical Earth model with $C = 40,003.2$ kilometers.

[7] North Atlantic Tracks (NATs) are relocated on a daily basis by the Shanwick and Gander Oceanic Centers. Depicted in the figure are NAT A,C, and E of September 8, 2009. Active NAT can be found at `https://www.notams.jcs.mil/common/nat.html`.

meridians rounded up to degrees of latitude. This allows to navigate along loxodromes between waypoints and still follow an approximated orthodrome.

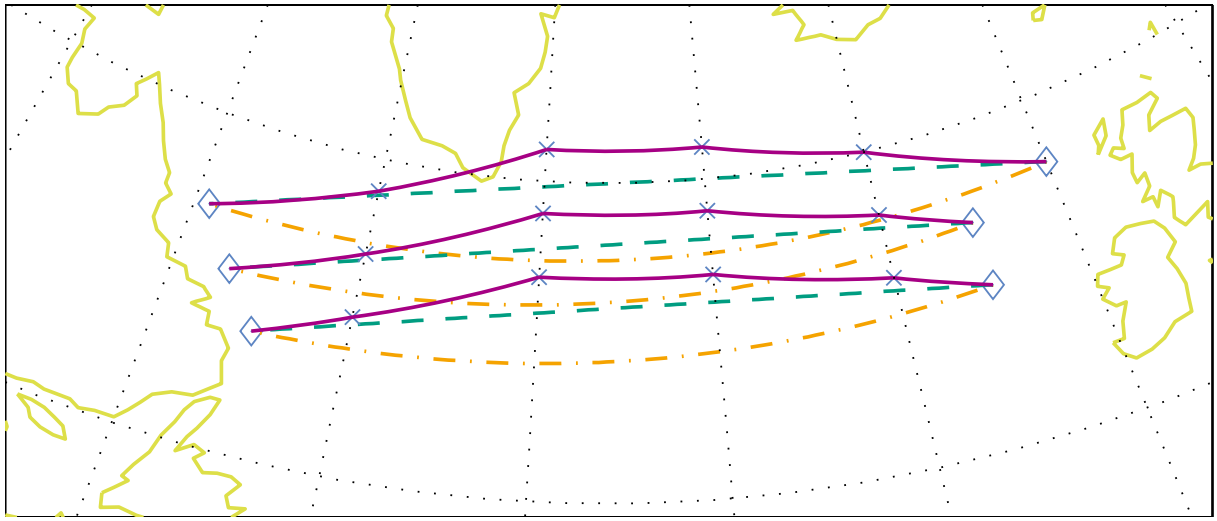


Figure 4.10.: North Atlantic Tracks in a gnomonic map projection. Depicted are the orthodromes (dashed) and loxodromes (dash-dotted) between the endpoints, and loxodromes between intermediate waypoints (solid).

For our mathematical operations, path approximations with linearly interpolated linestring segments are desirable for their computational simplicity. If a (loxodrome or orthodrome) path ℓ is approximated by a linestring ℓs with N_{approx} segments, the error introduced depends on the length of the segments or, provided segments have a constant or maximum length, the number of segments. With $N_{\text{approx}} \to \infty$, there would be no error.

Obviously, the deviation between the three path types depends on, firstly, the length of the path, secondly, its distance from the equator, and thirdly, its direction, i.e., the angle from one end point to the other, which is the loxodrome's bearing. The question is therefore how large an error is acceptable and, given a certain path, how long a segment of ℓs may be at most to stay within the error limit. At the time of writing this thesis, Chamberlain *et al.* from the National Aeronautics and Space Administration (NASA) Jet Propulsion Laboratory published a paper with "Some Algorithms for Polygons on a Sphere" [34]. They investigated similar issues as appear here, and answer the above question by relating the acceptable error (which they define as the lateral deviation from the "correct" path) e_{max} with a maximum length of an interpolation segment l_{max} for a path described by the two characteristics, latitude of the southwesterly point ϕ_P and bearing Φ. They argue that a closed-form expression relating e_{max}, l_{max}, ϕ_P, and Φ promises to be so complex that it provides no insight and instead provide tables that relate the variables. Specifying a maximum acceptable error, these tables can be used

to find out for every edge segment how long it may be at most. All edge segments that exceed this maximum length can be approximated with smaller segments. Equations to calculate the coordinates of the intermediate vertices are also provided by Chamberlain *et al.*.

Regarding the application in this thesis, we see that realistic lengths of polygon boundary segments used in aeronautical data will rarely exceed tens of nautical miles, usually much less. The path between New York City and Moscow that has been used as a running example in this section is thus not a realistic example for an airspace polygon boundary segment. As a real-world example, take the *Rhein UIR* that we used as example before[8]. It is quite large an airspace and located between the 47° and 52° parallel, thus probably containing edge segments with larger deviations between orthodrome and linear interpolation than most others. It is defined by 195 edge segments. In the case of this airspace, accepting a maximum error of 100 m lateral deviation, only five boundary segments exceed the maximum length and would have to be cut into shorter segments using one additional vertex each. Then the airspace polygon's boundary segments can simply be treated as linear interpolations of the vertices.

Assuming an error in the magnitude of 100 m is generally acceptable for our application, because air corrdidors are usually defined with a 5 nm lateral buffer for safety reasons. A 100 m error would thus make approximately 0.5 % of the width. The calculation of intermediate vertices for the few segments that are too long seems an acceptable preprocessing effort regarding the spared calculation complexity involved in handling loxodrome and orthodrome path types. This approach is therefore followed in the notification service. A global maximum lateral deviation error $e_{\max}$ is defined. When spatial filters or notifications are input at the publish/subscribe interface of the middleware, they are searched for too long boundary segments and additional vertices are inserted if required. Within the application itself, all geometric operations are handled in Euclidean space.

4.4 Spatiotemporal Filters

Now that interval and (geo-)spatial filters have been introduced, spatiotemporal filters can be defined. A subscription filter F for spatiotemporal notifications consists of right-open intervals over discrete point types $T_F = \left[t^-, t^+\right), V_F = \left[v^-, v^+\right)$ for the time and the vertical space of interest and of a simple polygon H_F representing the horizontal region of interest, $F = \left(H_E, V_E, T_E\right)$.

[8] Section 3.21, Figure 3.21 on Page 66

Figure 4.11 shows as an example a visualization of the filters thus constructed for the trajectory of the Frankfurt-Madrid flight from the introduction[9]. The trajectory is interpolated[10] and extended geometrically with a safety buffer, and spatiotemporal filters are defined for each leg. For the two horizontal spatial dimensions, the filter area is defined by polygons implementing spatial filters, whereas the altitude and time dimensions are represented by interval filters resulting in multidimensional boxes in the filterspace.

To match a spatiotemporal notification $n = (H_E, V_E, T_E)$, where H_E, V_E, T_E denote the horizontal and vertical spatial effectivity and the temporal effectivity of the event, respectively, the notification must affect some part of the subscription's horizontal region, at least one vertical space point and at least one chronon of the temporal interval.

Definition Spatiotemporal Notification Matching: *A spatiotemporal filter* $F = (H_F, V_F, T_F)$ *matches a spatiotemporal notification* $n = (H_E, V_E, T_E)$ *iff the horizontal regions intersect and the altitude intervals intersect and the temporal intervals intersect.*

$$F(n) \Leftrightarrow \text{intersect}(H_F, H_E) \wedge \text{intersect}(V_F, V_E) \wedge \text{intersect}(T_F, T_E)$$

Filter relationships too are defined as conjunctions of the respective relationships of interval and spatial filters.

Definition Spatiotemporal Filter Equality: *Two spatiotemporal filters* $F = (H_F, V_F, T_F), G = (H_G, V_G, T_G)$ *are equal iff* H_F *and* H_G *are equal and* V_F *and* V_G *are equal and* T_F *and* T_G *are equal.*

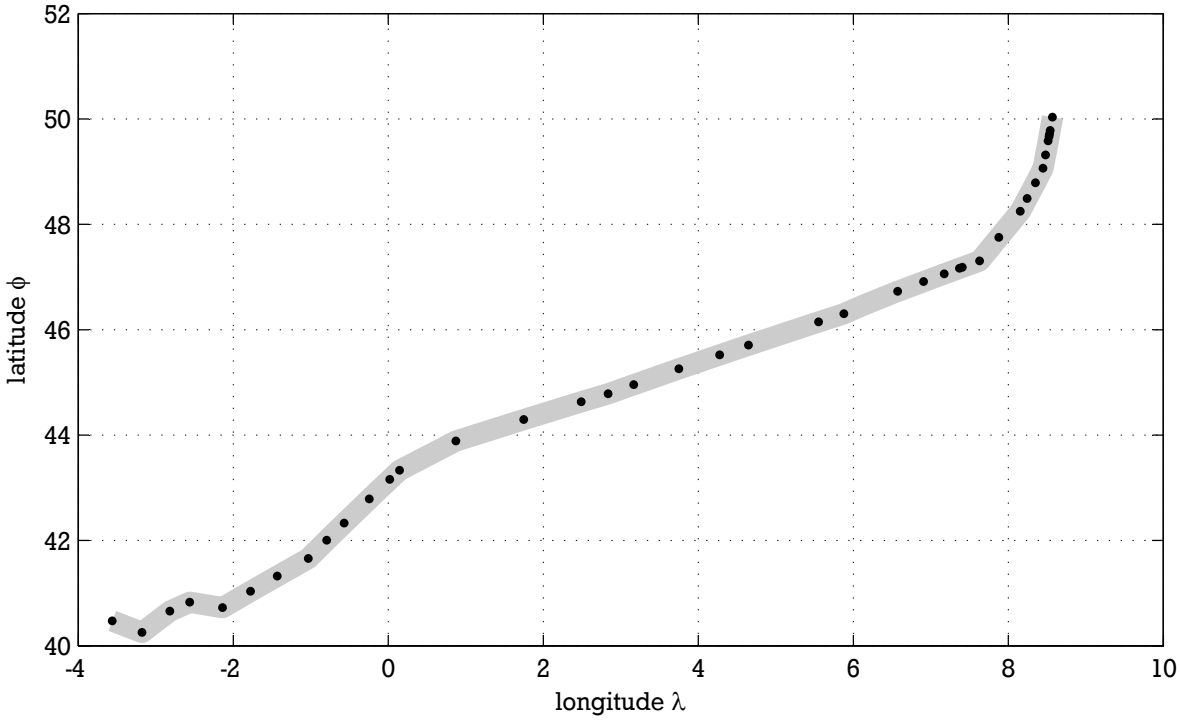
$$F \equiv G \Leftrightarrow \text{equal}(H_F, H_G) \wedge \text{equal}(V_F, V_G) \wedge \text{equal}(T_F, T_G)$$

Definition Spatiotemporal Filter Intersection: *Two spatiotemporal filters* $F = (H_F, V_F, T_F), G = (H_G, V_G, T_G)$ *are intersecting iff* H_F *and* H_G *intersect and* V_F *and* V_G *intersect and* T_F *and* T_G *intersect.*

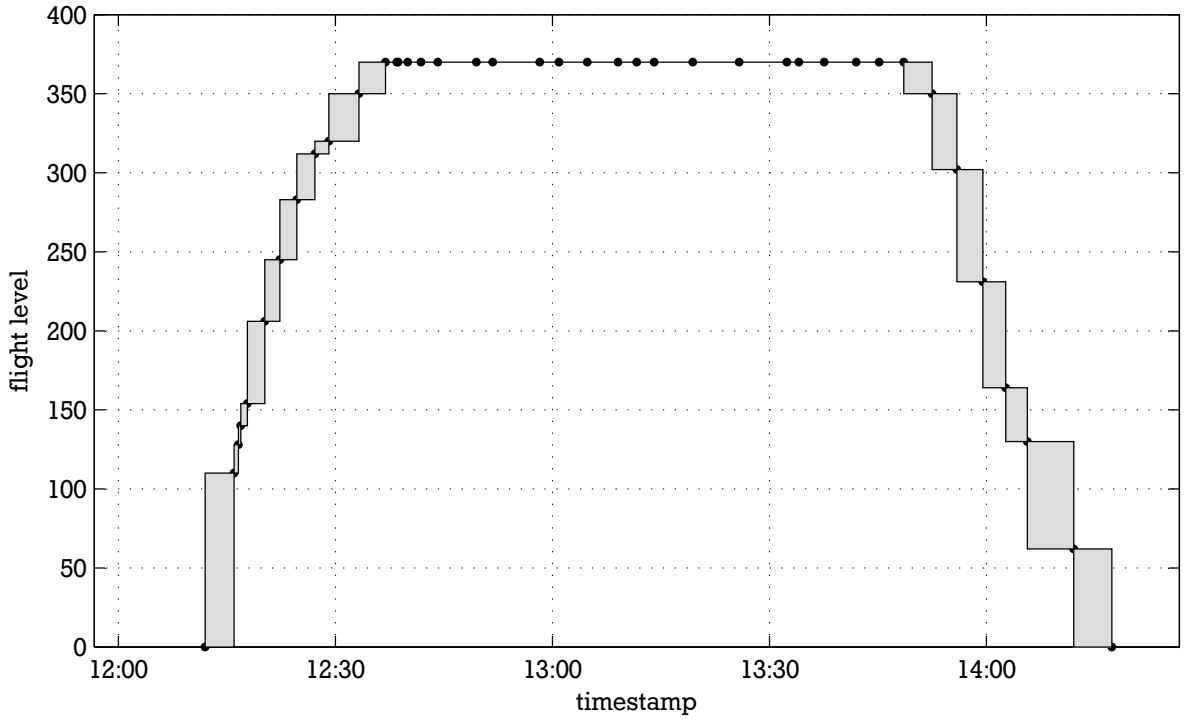
$$F \sqcap G \Leftrightarrow \text{intersect}(H_F, H_G) \wedge \text{intersect}(V_F, V_G) \wedge \text{intersect}(T_F, T_G)$$

[9] Section 1.1.2, Figure 1.4 on Page 7

[10] As discussed in the previous section, it is safe to apply linear interpolation if the legs are short enough, which can be assumed.



(a) longitude and latitude



(b) time and altitude

Figure 4.11.: Spatiotemporal filters for the 4D trajectory of a flight

Definition Spatiotemporal Filter Disjointness: *Two spatiotemporal filters* $F = (H_F, V_F, T_F), G = (H_G, V_G, T_G)$ *are disjoint iff* H_F *and* H_G *are disjoint or meet or* V_F *and* V_G *are disjoint or* T_F *and* T_G *are disjoint.*

$$F \not\between G \Leftrightarrow \text{disjoint}(H_F, H_G) \vee \text{meet}(H_F, H_G) \vee \text{disjoint}(V_F, V_G) \vee \text{disjoint}(T_F, T_G)$$

Definition Spatiotemporal Filter Cover: *A spatiotemporal filter* $F = (H_F, V_F, T_F)$ *covers a spatiotemporal filter* $G = (H_G, V_G, T_G)$ *iff* H_F *and* P_Q *are equal or* H_F *contains* H_G *or* H_F *covers* H_G *and* V_F *and* V_G *are equal or* V_F *includes* V_G *or* V_G *begins* V_F *or* V_G *ends* V_F *and* T_F *and* T_G *are equal or* T_F *includes* T_G *or* T_G *begins* T_F *or* T_G *ends* T_F. *F properly covers G if F covers G but* H_F *and* H_G *are not equal or* V_F *and* V_G *are not equal or* T_F *and* T_G *are not equal.*

$$\begin{aligned} F \sqsupseteq G \Leftrightarrow\ & (\text{equal}(H_F, H_G) \vee \text{contains}(H_F, H_G) \vee \text{covers}(H_G, H_F)) \\ & \wedge (\text{equal}(V_F, V_G) \vee \text{includes}(V_F, V_G) \vee \text{begins}(V_G, V_F) \vee \text{ends}(V_G, V_F)) \\ & \wedge (\text{equal}(T_F, T_G) \vee \text{includes}(T_F, T_G) \vee \text{begins}(T_G, T_F) \vee \text{ends}(T_G, T_F)) \\ F \sqsupset G \Leftrightarrow\ & F \sqsupseteq G \wedge (\neg\text{equal}(H_F, H_G) \vee \neg\text{equal}(V_F, V_G) \vee \neg\text{equal}(T_F, T_G)) \end{aligned}$$

5 Filter Merging

This Chapter discusses filter merging for each of the basic filter types introduced in the previous chapter and presents a formalization of the trade-off problem between filter quantity and filtering quality as well as an approach to imperfect filter merging and different merging strategies.

Section 5.1 introduces the merging operations and discusses conditions for perfect and imperfect merging of (N-)interval and spatial filters. We find that the computational complexity involved in spatial filter handling makes the application of perfect spatial filter mergers uneffective. Instead, an imperfect merging approach is taken, using envelopes of the filter polygon to build simplified imperfect spatial filter mergers, which take the form of 2-interval filters, thus leaving only multidimensional interval filters to be regarded in the further discussion of imperfect merging strategies.

While merging of two filters is always possible, the challenge is to decide when it is useful to merge filters. Section 5.2 formally states the problems to be solved and sketches a heuristic algorithm. The approach involves assessing the quality of mergers, which is discussed in detail in Section 5.3, where different approaches to merger quality estimation for 1-interval filters and N-interval filters are presented. Subsection 5.4 finally presents and discusses experimental results for the different merging strategies applied to various sets of subscription filters and notifications.

5.1 Filter Merging Operations

Whether perfect or imperfect mergers result from the merging operation $F \sqcup G$ for two (N-)interval filters or spatial filters F, G depends on the relationship of F and G, i.e., on the relationship of their intervals or regions. These conditions for perfect and imperfect mergers are discussed in the following, and for spatial filters, the computational complexity of the required geometrical operation is considered. We shall find that the effectiveness of spatial filter perfect merging is questionable, and simplified imperfect mergers propose to be much more beneficial.

5.1.1 1-Interval Filters

In order to describe interval filter mergers, *interval mergers* are defined first.

Definition Interval Merger: *An* interval merger I_M *of the intervals* $I_A = \left[a^-, a^+\right)$ *and* $I_B = \left[b^-, b^+\right)$*, denoted* $I_M = I_A \sqcup I_B$*, is an interval* $I_M = \left[m^-, m^+\right)$*, where the begin point*

m^- is the minimum of the begin points of I_A and I_B and the end point m^+ is the maximum of the end points of I_A and I_B.

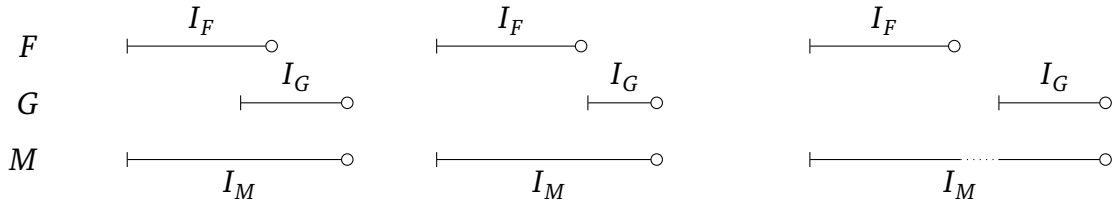
$$I_M = I_A \sqcup I_B = \left[\min(a^-, b^-), \max(a^+, b^+)\right) \tag{5.1}$$

The merger of two 1-interval filters is built from the interval merger of the intervals of the original filters.

Definition 1-Interval Filter Merger: *A merger $M = F \sqcup G$ of two 1-interval filters $F = (I_F), G = (I_G)$ is a 1-interval filter $M = (I_M)$, where I_M is the interval merger of I_F and I_G.*

$$M = F \sqcup G = (I_F \sqcup I_G)$$

M is a perfect merger iff I_F and I_G intersect (Figure 5.1a) or meet (b), because then the set of points in I_M is the union of the point sets of I_F and I_G. Otherwise, it is an imperfect merger, because I_M additionally includes the points between I_F and I_G (c).



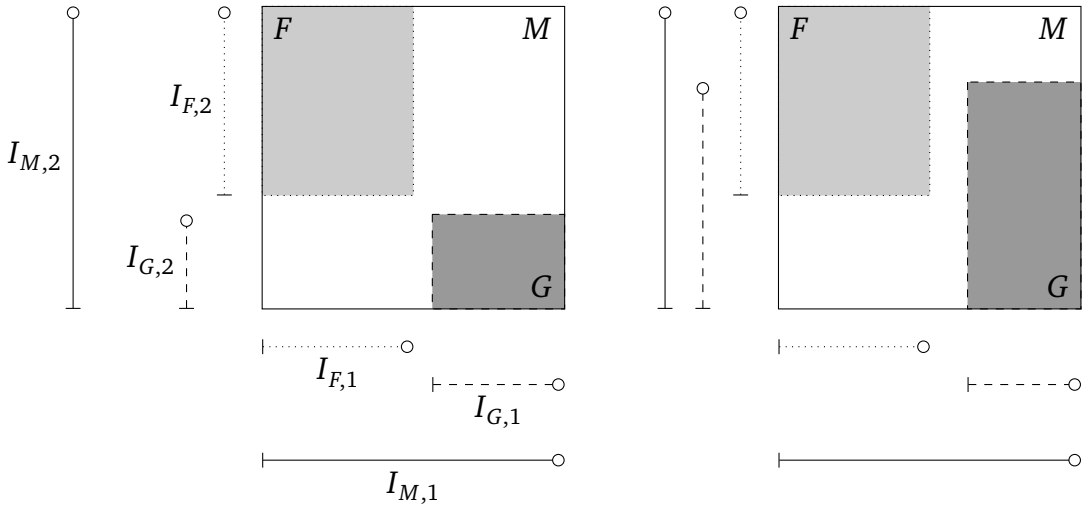
(a) I_F and I_G intersect **(b)** I_F and I_G meet **(c)** any other relation

Figure 5.1.: Perfect and imperfect merger M of interval filters F, G

5.1.2 N-Interval Filters

For N-interval filters, mergers are defined intuitively in line with 1-interval filters.

Definition N-Interval Filter Merger: *A merger M of two N-interval filters $F = (I_{F,i})_{i=1}^N, G = (I_{G,i})_{i=1}^N$, is an N-interval filter $M = (I_{M,i})_{i=1}^N$, where the intervals $I_{M,i}$ in each dimension i are the interval mergers of the respective intervals in F and G.*



(a) disjoint in both dimensions

(b) intersect in one dimension, disjoint in one dimension

(c) intersect in both dimensions

(d) equal in one dimension, intersect in one dimension

Figure 5.2.: Arrangements of 2-interval filters and resulting mergers.

$$M = F \sqcup G = \left(I_{F,i} \sqcup I_{G,i}\right)_{i=1}^{N}$$

Consider the relationships of intervals $I_{F,i}$ and $I_{G,i}$ and the implication for perfect mergers in the examples in Figure 5.2. For $M = F \sqcup G$ to be a perfect merger it is not sufficient that $\mathit{intersect}(I_{F,i}, I_{G,i})$ holds in every dimension (c). Instead, the intervals must be equal in all but one dimension, and in that dimension, they must intersect or meet (d). In all other cases, an imperfect merger results (a) (b).

5.1.3 Spatial Filters

Merging two spatial filters $F = (P_F), G = (P_G)$ obviously requires uniting the polygons P_F and P_G. This is however possible only if P_F and P_G intersect or meet. The resulting spatial filter is always a perfect merger.

Definition Spatial Filter Perfect Mergers: *Let $F = (P_F), G = (P_G)$ be spatial filters. If $\mathit{intersect}(P_F, P_G)$, the spatial filter $M = (P_M)$ with P_M being the polygon union of P_F and P_G is the perfect merger of F and G.*

$$M = F \sqcup G = (P_F \cup P_G) \text{ if } \mathit{intersect}(P_F, P_G)$$

The polygon union U of two polygons P,Q, $U = P \cup Q$, is a computationally expensive operation. It can be derived from the DCEL (see Section 4.2.3), which has to be computed for this purpose[1] The resulting polygon's complexity u is at least $\max(p,q)$. If P and Q are equal or P contains or covers Q or vice versa, $u = \max(p,q)$. In the worst case, $u > p+q$ as in the example in Figure 5.3. Since the cost of spatial filter matching, which requires a polygon intersection test, depends linearly on the complexity of the filter polygon, no efficiency benefit would be achieved by this merging.

Instead, spatial filter simplification can be applied by using the envelopes of the involved polygons. An imperfect merger of the spatial filters $F = (P_F), G = (P_G)$ is thus built as a 2-interval filter from the envelopes of the polygons P_F, P_G (Figure 5.4). The intervals of the merger $M = (I_{M,1}, I_{M,2})$ are the interval mergers of the envelope intervals of the polygons P_F, P_G. By first simplifying the spatial filters to 2-interval filters (c), the costly calculation of the polygon union $P_F \cup P_G$ (b) is avoided. It is obvious that $I_{M,1} = I_{F,1} \sqcup I_{G,1}$ and $I_{M,2} = I_{F,2} \sqcup I_{G,2}$ (d), which holds irregardless of the topological spatial relation of P_F and P_G due to the definition of interval mergers, which include

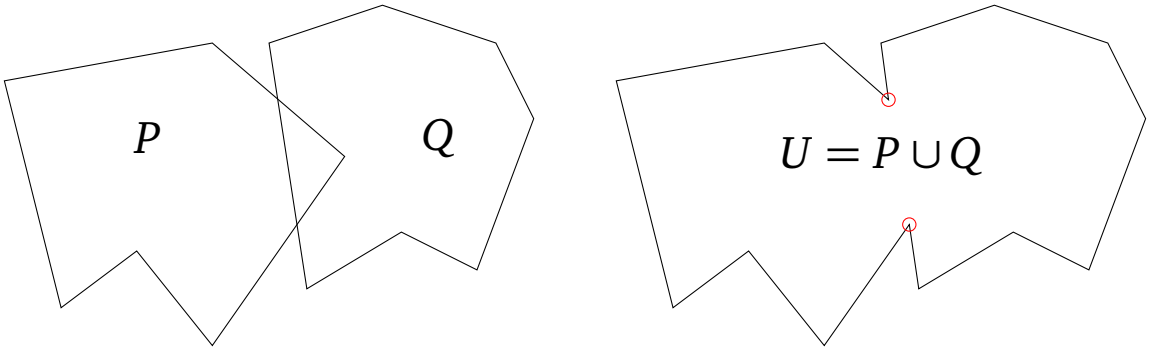
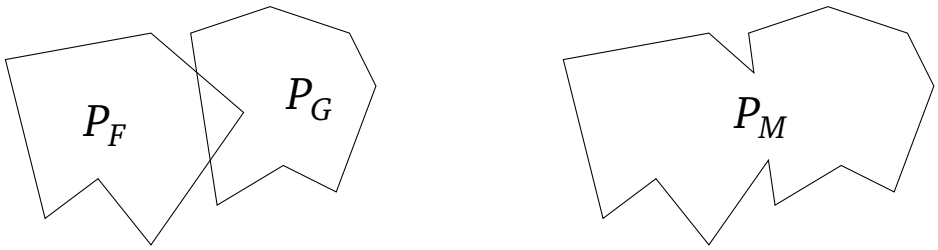
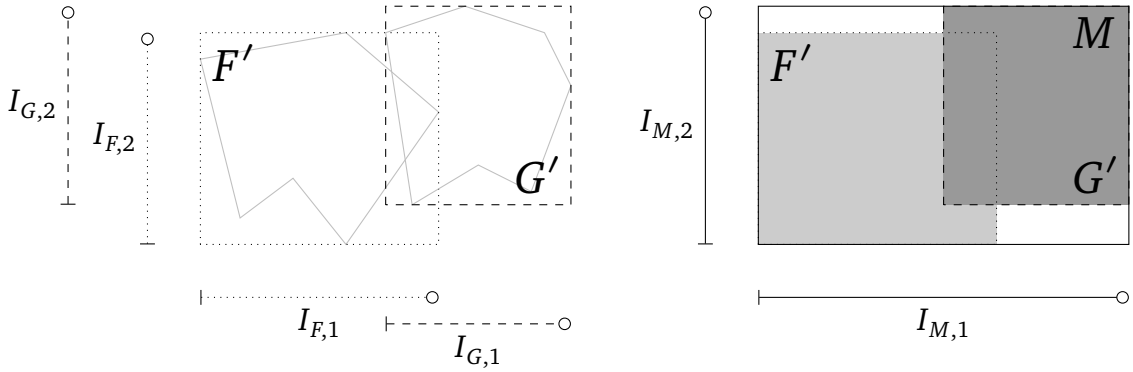


Figure 5.3.: Building the union of two polygons $P \cup Q$. In the worst case, the union has a higher complexity than the added complexity of P and Q.



(a) Polygons P_F, P_G of spatial filters F, G

(b) Polygon P_M of perfect merger filter $M = F \sqcup G$



(c) Simplification of spatial filters F, G to 2-interval filters F', G'

(d) Intervals $I_{M,1}, I_{M,2}$ of imperfect merger filter $M = F' \sqcup G'$

Figure 5.4.: Spatial filter perfect and imperfect merging

the possible space between disjoint intervals. Building such imperfect mergers is thus possible for any two or more spatial filters.

Definition Spatial Filter Imperfect Mergers: *An imperfect merger M of two spatial filters $F = (P_F)$ and $G = (P_G)$ is a 2-interval filter $M = \left(I_{M,1}, I_{M,2}\right)$, where $I_{M,1}, I_{M,2}$ are the interval mergers of the respective intervals from the envelopes of P_F, P_G.*

$$M = F' \sqcup G' = \left(I_{F,1} \sqcup I_{G,1}, I_{F,2} \sqcup I_{G,2}\right)$$
$$\text{where } \left(I_{F,1}, I_{F,2}\right) = \text{envelope}\left(P_F\right), \left(I_{G,1}, I_{G,2}\right) = \text{envelope}\left(P_G\right)$$

While the envelope operations theoretically require $O(p)$ and $O(q)$ time, we assume that the envelope is always computed and stored with a spatial filter when the filter is initially created. Then the imperfect merging operation can be applied in constant time.

Filter merging is consequently only applied on the simplified 2-interval filters. The merging strategies that are discussed in the next sections therefore exclusively treat (N-)interval filters.

5.2 Merging Problem Formalization

The imperfect filters that may result from the merging operations introduce the possibility of false positives leading to unnecessarily forwarded notifications, whereas the benefit is the reduction of filter numbers to handle. The goal of any merging strategy is therefore to find a good trade-off between routing table size *reduction* and matching *precision*.

The following Subsection 5.2.1 formally describes these goals preparing the problem statement of merging in Subsection 5.2.2. Subsection 5.2.3 presents the outline of a heuristical algorithm for finding a solution for the merging problem.

5.2.1 Merging Goals: Reduction and Precision

Filter merging can generally be applied on the set of subscription filters in a broker's routing table with the same destination, i.e., the subscription filters the broker has received from the same neighbor broker. Let $\mathscr{F}$ be such a set of subscription filters

[1] Note that the polygon intersection test required to decide whether it is at all possible to perfectly merge the filters takes only $O(2N)$ time in the worst case.

F_i, $\mathscr{F} = \{F_i\}$. Applying a merging strategy, this set $\mathscr{F}$ is reduced to a smaller set $\mathscr{F}'$ consisting partially or entirely of filter mergers. The achieved *reduction* r is

$$r\left(\mathscr{F}'\right) = \frac{\left|\mathscr{F}\right| - \left|\mathscr{F}'\right|}{\left|\mathscr{F}\right|}, \tag{5.2}$$

where $|\mathscr{F}|$ denotes the cardinality of $\mathscr{F}$. Furthermore, let $N(\mathscr{F})$ denote the set of notifications matched by any of the filters in $\mathscr{F}$, $N(\mathscr{F}) = \cup N(F_i) = \{n | \exists F_i \in \mathscr{F}.F(n)\}$. The reduced set $\mathscr{F}'$ is required to match all notifications that are matched by the original set $\mathscr{F}$, $N\left(\mathscr{F}'\right) \supseteq N\left(\mathscr{F}\right)$, i.e., false negatives are disallowed. With imperfect mergers however, $\mathscr{F}'$ possibly matches more notifications than $\mathscr{F}$. Then $N\left(\mathscr{F}'\right)$ contains false positives, namely those in $N\left(\mathscr{F}'\right) \setminus N\left(\mathscr{F}\right)$. The matching precision p is given by the number of true positives divided by the number of all positives:

$$p\left(\mathscr{F}'\right) = \frac{|N(\mathscr{F})|}{|N(\mathscr{F})| + |N(\mathscr{F}') \setminus N(\mathscr{F})|} = \frac{|N(\mathscr{F})|}{|N(\mathscr{F}')|}$$

To determine the precision experimentally, we use a sample set of notifications $\mathscr{N}$. It is then given by

$$p_{\mathscr{N}}\left(\mathscr{F}'\right) = \frac{|\mathscr{N} \cap N(\mathscr{F})|}{|\mathscr{N} \cap N(\mathscr{F}')|}, \tag{5.3}$$

where $\mathscr{N} \cap N(\mathscr{F})$ is the set of all notifications in $\mathscr{N}$ that are matched by some filter in $\mathscr{F}$, $\mathscr{N} \cap N(\mathscr{F}) = \{n \in \mathscr{N} | \exists F \in \mathscr{F}.F(n) = true\}$.

5.2.2 Filter Merging Problem Statement

The goal of any merging strategy is high precision at high reduction. Since higher reduction potentially results in lower precision, these values can only be balanced toward application requirements, aiming to maximize one value with the other value being constant. Hence, the general problem statement of filter merging is:

> General Filter Merging Problem (GFMP)*: Given a set of filters* $\mathscr{F}$ *and a* reduction goal r^* *[*precision goal p^**], find a set of filters* $\mathscr{F}'$ *with* $r_{\mathscr{F}}(\mathscr{F}') \geq r^*$ *[*$p_{\mathscr{F}}(\mathscr{F}') \geq p^*$*] and* $N(\mathscr{F}) \setminus N(\mathscr{F}') = \emptyset$*, where* $p_{\mathscr{F}}(\mathscr{F}')$ *[*$r_{\mathscr{F}}(\mathscr{F}')$*] is maximal.*

A. Crespo *et al.* have described a similar problem called *Query Merging Problem (QMP)* and have shown that it is NP-hard [43]. The goal of QMP is, given a query set, to find a minimal query set that returns the same results as the original query set. An algorithm

for GFMP can also solve QMP by setting the precision goal in GFMP to $p^* = 1$, thus disallowing false positives. Hence, solving GFMP optimally is NP-hard as well.

The problem can be stated slightly differently taking into account the actual problem for a broker at runtime, where subscriptions are to be processed upon reception: Whenever a new subscription filter G is received, the best merging candidate $F_{\text{mc}} \in \mathscr{F}$ for G is to be identified.

> N-to-1 Filter Merging Problem (N1FMP)*: Given a set of filters $\mathscr{F}$, a filter G, and a* reduction goal r^* *[*precision goal p^**], find the merging candidate filter $F_{\text{mc}} \in \mathscr{F}$ such that $r_{\mathscr{F}\cup\{G\}}(\mathscr{F}') \geq r^*$ $[p_{\mathscr{F}\cup\{G\}}(\mathscr{F}') \geq p^*]$ and $r_{\mathscr{F}\cup\{G\}}(\mathscr{F}')$ $[p_{\mathscr{F}\cup\{G\}}(\mathscr{F}')]$ is maximal. The set $\mathscr{F}'$ is the resulting set when merging F_{mc} and G within $\mathscr{F}$, $\mathscr{F}' = \mathscr{F} \setminus \{F_{\text{mc}}\} \cup (F_{\text{mc}} \sqcup G)$*

One can assume that the filter set resulting after adding a number of filters in N1FMP will not yield the optimal solution, because then an optimal solution for N1FMP would be an optimal solution for GFMP. Instead of aiming for the optimal solution, the approach taken here is to develop a heuristic strategy, which finds a "good" solution.

Input: Set of filters $\mathscr{F}$, single filter G
Output: Set of filters $\mathscr{F}'$

$F_{\text{mc}} \leftarrow$ `findMergingCandidate(`$\mathscr{F}$`,`G`)`
if $F_{\text{mc}} = null$ **then**
    $\mathscr{F}' \leftarrow \mathscr{F} \cup \{G\}$
end
else
    $M \leftarrow F_{\text{mc}} \sqcup G$
    $\mathscr{F}' \leftarrow \mathscr{F} \cup \{M\} \setminus \{F_{\text{mc}}\}$
end
return $\mathscr{F}'$

Figure 5.5.: Algorithm frame for solving N1FMP

Figure 5.5 presents an outline of an alogrithm to solve N1FMP. It involves the function *findMergingCandidate*, for which two alternative implementations are discussed next.

5.2.3 Implementation of a Heuristical Approach

The search for a merging candidate in *findMergingCandidate* can be implemented in two alternative ways, either by checking all filters in $\mathscr{F}$ and selecting the best one, or by checking one by one until a "good enough" merging candidate is found. Either way, it

must be determined how good a merging candidate for G a given filter is. The approach taken here is to quantitatively estimate how good or bad the mergers $M_i = F_i \sqcup G$ for filters $F_i \in \mathscr{F}$ are using a *merging penalty score* function $MP : \mathbb{F}^2 \to \mathbb{R}$ (where $\mathbb{F}$ is the set of all filters) that returns a numerical value as an estimate of the quality of the merger $F \sqcup G$. The lower $MP(F, G)$ the better the merger $M = F \sqcup G$ is.

In the exhaustive approach, $MP(F, G)$ is evaluated for every $F \in \mathscr{F}$. The merging candidate F_{mc} is the filter F where $MP(F, G)$ is minimal. For this approach to work, a threshold value p_0 is needed, which defines the minimal merging penalty value, for which merging will take place. Without this condition, every newly added filter would be merged with some filter, however bad the mergers are, and, as a prohibitive consequence, $\mathscr{F}$ would always contain only one filter. F_{mc} is returned as a suitable merging candidate only if $MP\big(F_{\text{mc}}, G\big) \leq p_0$. The implementation is shown in Function *findMergingCandidateExhaustive*. The search obviously runs in $O(|\mathscr{F}|)$ time (best, worst, and average case), because it needs to visit every filter $F \in \mathscr{F}$ exactly once.

Input: Set of filters $\mathscr{F}$, single filter G, merging penalty score threshold p_0
Output: Merging candidate F_{mc}

$p_{\text{min}} \leftarrow +\infty$; `// minimum merging penalty value`
$F_{\text{mc}} \leftarrow$ null; `// merging candidate`

foreach $F \in \mathscr{F}$ **do**
    if `MP`$(F, G) < p_{\text{min}}$ **then**
        $p_{\text{min}} \leftarrow$ `MP`(F, G)
        $F_{\text{mc}} \leftarrow F$
    end
end
if $p_{\text{min}} \leq p_0$ **then**
    return F_{mc}
end
return *null*

Function `findMergingCandidateExhaustive(`$\mathscr{F}$`, `G`)`

Alternatively, the penalty score threshold p_0 is used already in the phase of finding a merging candidate. While iterating through the filters $F \in \mathscr{F}$, the merging candidate F_{mc} is taken to be the first found filter where $MP(F, G) \leq p_0$ (Function *findMergingCandidateNonExhaustive*).

Obviously, this procedure also runs in $O(|\mathscr{F}|)$ in the worst case, i.e., when no good enough merging candidate is found. However, in the best case the first checked filter is a suitable merging candidate. The average performance is slightly better than the worst case, because in all the cases where a merging candidate is found, the function runs

Input: Set of filters $\mathscr{F}$, single filter G, merging penalty score threshold p_0
Output: Merging candidate F_{mc}

foreach $F \in \mathscr{F}$ **do**
  if $\texttt{MP}(F, G) \leq p_0$ **then**
    return F
  end
end
return *null*

Function `findMergingCandidateNonExhaustive(`$\mathscr{F}$`, `G`)`

in $O(|\mathscr{F}|/2)$ average time. Although this is the case only for few instances, the overall average performance can still be expected to be better than the one of the exhaustive candidate search. The downside is that the filter F that G is merged with may not be the best merging candidate.

The experimental results presented in Subsection 5.4 however show that the difference is —somewhat contradicting the intuition— negligible, with respect to the achievable precision as an indicator of how good the selected merging candidates are, as well as to performance. Possible reasons are discussed there.

5.3 Filter Merger Quality Estimation

A merging penalty score function $MP(F, G)$ shall evaluate how beneficial or unbeneficial a merger $F \sqcup G$ would be. In Subsection 5.3.1, the approach to merger quality estimation is first discussed informally on the basis of an example set of five 1-interval merging candidates. To formally approach the quality estimation of mergers, i.e., derive statements on how beneficial a merger of two filters can be expected to be, characteristics of filters and filter sets are introduced in Subsection 5.3.2.

The overall goal —maximizing reduction and precision of a filter set— translates for the situation of quality estimation of a single merger filter to evaluating how much the creation of the merger would negatively impact the overall precision. This should be represented in the result of the penalty score functions. We propose two general approaches to penalty score functions with individual variants, based on filter size and and filter distance. The size-based approach introduced in Subsection 5.3.3 puts the sizes of the original filters in relation with the merger size, assuming that the smaller the merger the less it reduces filter quality. Subsection 5.3.4 presents a variant of the general approach based on the sizes of the interval mergers in each filter space dimension. The distance-based approach introduced in Subsections 5.3.5 assumes that the closer two filters are in the filter space the better the resulting merger is.

We find that the proposed basic approaches exhibit a drawback that could lead to unsatisfactory results, namely that all filter space dimensions are treated equally while they may have widely varying characteristics. In Subsection 5.3.6, the concept of a virtual filter space with normalized metrics among the individual dimensions is introduced, and an enhanced (normalized) distance-based merging pernalty score function is proposed.

5.3.1 Merger Quality

"Good" mergers are those that do not (or, to a small extent) reduce overall filter quality, expressed in the precision value for a set of filters. The general approach taken here to estimate the merger quality is the assumption that good mergers result from merging filters that are "similar". Similar filters are those that match to a large degree the same notifications, i.e., where $|N(F) \cap N(G)|$ is large.

As an example, let $\mathscr{F} = \{F_i\}_{i=1}^{5}$ of the 1-interval filters in Figure 5.8. Say, the best merging candidate $F_{\text{mc}} \in \mathscr{F}$ for G is to be found. The filters M_{i} are the potential mergers, $M_{\text{i}} = G \sqcup F_{\text{i}}$.

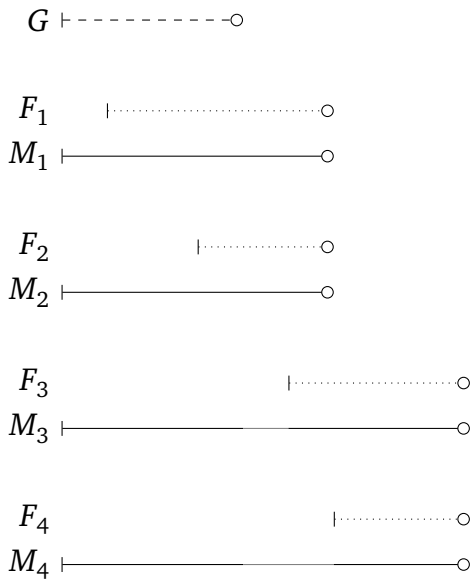


Figure 5.8.: Comparison of 1-interval mergers

M_1 and M_2 are both perfect mergers, but F_1 seems to be the better merging candidate, because the overlap with G is larger than for F_2, hence $\left|N(F_1) \cap N(G)\right| > \left|N(F_2) \cap N(G)\right|$. M_3 and M_4 are imperfect mergers because both include points not included in either of the original filters. False positives result from notifications being matched by these points. The number of these points is smaller in M_3, therefore F_3 still seems to be the better merging candidate than F_4. This assessment is based on the assumption

that notifications are distributed uniformly in the interval domain. If it were known, for instance, that no notification ever concerns the *imperfection* of M_3 and M_4, i.e., the interval additionally matched by M_3 and M_4, both would be equally good merging candidates. However, in the absence of contrary information, it seems reasonable to assume a uniform distribution of notifications in the filter space.

5.3.2 Filter, Notification, and Filter Space Characteristics

The characteristics of filters, notifications, and sets of these that are formally introduced in the following serve to derive useful merging strategies and provide the basis for the explanation of the experimental results described later.

We have thus far only informally mentioned the *filter space* at various occasions. We start by describing it formally, then define filter and notification *size* and *distance*, and finally introduce the concept of filter or notification *density* in the filter space.

Filter Space

N-interval filters $F = \left(I_{F,i}\right)_{i=1}^{N}$ and notifications $n = \left(I_{E,i}\right)_{i=1}^{N}$ are defined by intervals over discrete point types. Each interval domain is isomorph to $\mathbb{N}$, i.e., a 1-interval filter or notification can be respresented by an interval in $\mathbb{N}$ such that every point included in the interval maps to a natural number. Consequently, N-interval filters and notifications can be represented by an N-dimensional box (an N-polytope) in $\mathbb{N}^N$.

The begin and end points $f_1^-, f_1^+, f_2^-, f_2^+, \ldots, f_N^-, f_N^+$ of the intervals in each dimension define the 2^N corner points $\mathbf{p_i^F}$ of this polytope: $\mathbf{p_1^F} = (f_1^-, f_2^-, \ldots, f_N^-), \mathbf{p_2^F} = (f_1^+, f_2^-, \ldots, f_N^-), \mathbf{p_3^F} = (f_1^-, f_2^+, \ldots, f_N^-), \mathbf{p_4^F} = (f_1^+, f_2^+, \ldots, f_N^-), \ldots, \mathbf{p_{2^N}^F} = (f_1^+, f_2^+, \ldots, f_N^+)$.

Size

Let $I = \left[i^-, i^+\right)$. The size of the interval I, denoted $|I|$, is the number of values $i \in I$, i.e., the number of discrete points contained in I. For instance, $|I| = 0$ iff $i^- = i^+$. Based on this, we define the size of interval filters and notifications as follows.

Definition Filter and Notification Size: *The size $|F|$ or $|n|$ of an N-interval filter $F = \left(I_{F,i}\right)_{i=1}^{N}$ or notification $n = \left(I_{E,i}\right)_{i=1}^{N}$ is the product of the sizes of its intervals.*

$$
\begin{aligned}
|F| &= \prod_{i=1}^{N} \left| I_{F,i} \right| \\
|n| &= \prod_{i=1}^{N} \left| I_{E,i} \right| .
\end{aligned}
\tag{5.4}
$$

Regarding its polytope representation in $\mathbb{N}^N$, the size of a filter or notification is the number of discrete points inside and on the edges and corners of the polytope.

Given a sample set of filters $\mathscr{F}$ or notifications $\mathscr{N}$, the *mean size* $\overline{|F|}$ or $\overline{|n|}$ of filters $F \in \mathscr{F}$ and notifications $n \in \mathscr{N}$ can be calculated:

$$
\begin{aligned}
\overline{|F|} &= \frac{1}{|\mathscr{F}|} \cdot \sum_{F \in \mathscr{F}} |F| \\
\overline{|n|} &= \frac{1}{|\mathscr{N}|} \cdot \sum_{n \in \mathscr{N}} |n|.
\end{aligned}
\tag{5.5}
$$

Distance

To formally define a distance metric for interval filters and notifications, we first introduce a shorthand notation for elements of intervals. Let $I = \left[i^-, i^+ \right) = \left\{ i^-, i^- + 1, i^- + 1 + 1, \ldots, i^+ - 1 \right\}$ and let i be some point in I, $i \in I$. The successor of i is $i + 1$. We then write shorthand $i + 2$ for the successor of $i + 1$, $i + 2 = i + 1 + 1$. Similarly, $i + 3 = i + 1 + 1 + 1$ and so on. Another expression for the end point i^+ of the interval I is then $i^- + |I| - 1$.

Using this notation, we can define the *median* $\tilde{\imath}$ of the interval I:

$$
\tilde{\imath} = \begin{cases} i^- + \frac{1}{2}(|I| - 1) & |I| \text{ odd} \\ i^- + \frac{1}{2}|I| - 1 & |I| \text{ even} \end{cases}
\tag{5.6}
$$

The *centroid* of a filter or notification can now be defined on the basis of the median of the filter intervals.

Definition Filter and Notification Centroid: *The centroid $\tilde{f}$ of an N-interval filter $F = \left(I_{F,i} \right)_{i=1}^{N}, I_{F,i} = \left[f_i^-, f_i^+ \right)$ is the point $\tilde{f} = \left(\tilde{f}_i \right)_{i=1}^{N}$, where $\tilde{f}_i$ is the median of the interval $I_{F,i}$, i.e., the point $f_i^- + \frac{|I_{F,i}|-1}{2}$ if the number of points in $I_{F,i}$ is odd, and $f_i^- + \frac{|I_{F,i}|}{2} - 1$ otherwise. The centroid $\tilde{e}$ of an N-interval notification $n = \left(I_{E,i} \right)_{i=1}^{N}, I_{E,i} = \left[e_i^-, e_i^+ \right)$ is defined accordingly.*

The centroid's representation in the filter space is the vector $\tilde{\mathbf{f}} = (\tilde{f}_1, \tilde{f}_2, \ldots, \tilde{f}_N)$ that is closest to the geometric center of the polytope representing the filter or notification, e.g., $\tilde{\mathbf{f}} = round\left(\frac{1}{2^N}\sum_{i=1}^{2^N} \mathbf{p_i^F}\right)$.

The distance d of two filters or notifications can now be defined as the distance between their centroids. Regarding an appropriate distance measure, the intuitive Euclidean distance obviously cannot be used, because the filter space is not isomorph to Euclidean space $\mathbb{R}^N$. Instead, we use a variant of L_1 distance, also known as Manhattan or taxicab distance [116]:

Definition Filter and Notification Distance: *The distance d of two N-interval filters $F = \left(I_{F,i}\right)_{i=1}^{N}, G = \left(I_{G,i}\right)_{i=1}^{N}$ is the sum of the distances of the medians $\tilde{f}_i$ and $\tilde{g}_i$ of the intervals $I_{F,i}$ and $I_{G,i}$, i.e., the number of points between $\tilde{f}_i$ and $\tilde{g}_i$, over all N dimensions. Assume $\tilde{f}_i \leq \tilde{g}_i$ without loss of generality. Then the distance of $\tilde{f}_i$ and $\tilde{g}_i$ is the size of the interval $\left[\tilde{f}_i, \tilde{g}_i\right)$. The distance d of two N-interval notifications m, n is defined accordingly as the sum of the distances of the intervals' medians $\tilde{m}_i, \tilde{n}_i$ over all dimensions.*

$$
\begin{aligned}
d(F,G) &= \left|\left|\tilde{f} - \tilde{g}\right|\right| = \sum_{i=1}^{N} \left| \left[\min\left(\tilde{f}_i, \tilde{g}_i\right), \max\left(\tilde{f}_i, \tilde{g}_i\right) \right) \right| \\
d(m,n) &= ||\tilde{m} - \tilde{n}|| = \sum_{i=1}^{N} \left| \left[\min\left(\tilde{m}_i, \tilde{n}_i\right), \max\left(\tilde{m}_i, \tilde{n}_i\right) \right) \right|
\end{aligned}
\tag{5.7}
$$

In the filter space representation in $\mathbb{N}^N$, the distance is simply the common L_1 distance d_1 between the representations of the centroids, e.g., $d(F,G) = d_1\left(\tilde{\mathbf{f}}, \tilde{\mathbf{g}}\right) = \left|\tilde{\mathbf{f}} - \tilde{\mathbf{g}}\right|_1 = \sum_{i=1}^{n} \left|\tilde{f}_i - \tilde{g}_i\right|$

In the case of 1-interval filters or notifications, the distance is obviously simply the number of points between the medians of the intervals. If $F = \left(I_F\right)$ and $G = \left(I_G\right)$ then $d(F,G) = \left|\left[\tilde{f}_i, \tilde{g}_i\right)\right|$ assuming $\tilde{f}_i \leq \tilde{g}_i$.

For sets of 1-interval filters $\mathscr{F}$ or notifications $\mathscr{N}$, we define the *mean distance* $\overline{d}(\mathscr{F})$ or $\overline{d}(\mathscr{N})$ as follows. Assume an ordered set of all $F \in \mathscr{F}$, where the order is given by the centroids $\tilde{f}$. Let F_i denote the i-th filter in the ordered set. Then the mean distance is the arithmetic mean of the distance between a filter and its successor in the ordered set (mean notification distance accordingly):

$$
\begin{aligned}
\overline{d}(\mathscr{F}) &= \frac{1}{|\mathscr{F}| - 1} \sum_{i=1}^{|\mathscr{F}|-1} d\left(F_i, F_{i+1}\right) \\
\overline{d}(\mathscr{N}) &= \frac{1}{|\mathscr{N}| - 1} \sum_{i=1}^{|\mathscr{N}|-1} d\left(n_i, n_{i+1}\right)
\end{aligned}
\tag{5.8}
$$

Note that a similar approach to defining the mean distance of N-interval filters or notifications in a set is not easiliy realizable, because the approach assumes that after ordering the set, the two neighbors of an element are the element's *nearest neighbors* in the whole set. For spaces with more than one dimension, no such ordering exists. Instead, the problem of ordering the elements such that the total distance over all elements is minimal is the *Traveling Salesman Problem*, which is known to be NP-equivalent [177].

However, for N-interval filters or notifications, the mean distance can be stated for each dimension individually by the above approach. We then denote with $\overline{d}_1(\mathscr{F})$ the mean distance of filters $F \in \mathscr{F}$ with respect to dimension 1 only, other dimensions and notification sets analogously.

Filter Space Density

Given a set of filters $\mathscr{F}$ (or set of notifications $\mathscr{N}$), we can define the filter or notification *density* of the filter space with respect to the sample set as follows.

Assuming the filter space is bounded, i.e., the filter space consists of a finite number of discrete points and is hence not isomorph to $\mathbb{N}^N$ but to a polytope in $\mathbb{N}^N$, let $|\mathbb{F}|$ denote the size of this polytope. It is – as with filters and notifications – the number of discrete points the polytope consists of. This is the *filter space size*. Then a filter of size $|F|$ matches a specific point of the filter space with a probability of $|F|/|\mathbb{F}|$. Using the mean filter size $\overline{|F|}$, we define the filter density in the filter space with respect to $\mathscr{F}$ as the average number of filters $F \in \mathscr{F}$ that match one filter space point. It is given by

$$\text{dens}(\mathscr{F}) = |\mathscr{F}| \frac{\overline{|F|}}{|\mathbb{F}|}. \tag{5.9}$$

Accordingly, we define the notification density with respect to a set of notifications $\mathscr{N}$, $dens(\mathscr{N})$, as the average number of notifications that affect any filter space point. It is calculated analogously.

In the case of the unbounded filter space $\mathbb{N}^N$, we can derive a density estimation using the mean size and mean distance of filters or notifications in the sample set. We regard the one-dimensional case only, i.e., we assume a one-dimensional filter space, and can apply the results later to derive density estimates for each dimension of the filter space individually.

Then the filter density in the filter space with respect to the set $\mathscr{F}$ is given by the mean size divided by the mean distance:

$$\text{dens}(\mathscr{F}) = \overline{|F|}/\overline{d}(\mathscr{F}). \tag{5.10}$$

The correctness can be seen by imagining an even distribution of filters, each of which has size $\overline{|F|}$, in the filter space dimension. If they were placed at distance $\overline{d}$ and such that no gaps remained, $\overline{|F|}/\overline{d}(\mathscr{F})$ filters would have to be "stacked" at every filter space point (Figure 5.9).

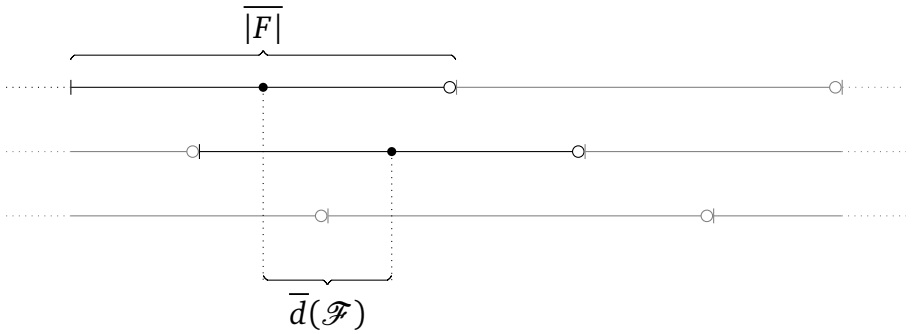


Figure 5.9.: 1-interval filter set density estimation. On average, $dens(\mathscr{F}) = 3$ filters match any filter space point.

Accordingly, the mean distance of notifications in a set of notifications, $\overline{d}(\mathscr{N})$, can be calculated, and, using the mean notification size $\overline{|n|}$, the notification density $dens(\mathscr{N})$ can be estimated.

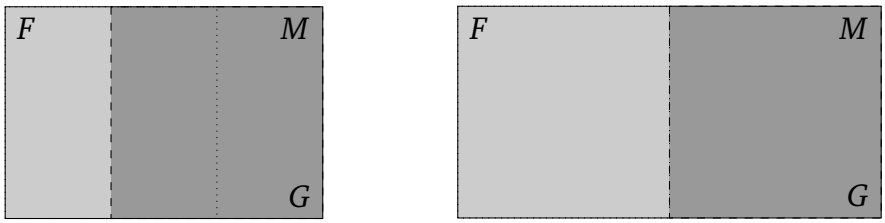
5.3.3 Size-based Merging Penalty

If a uniform distribution of notifications in the filter space is assumed, the size of the set of notifications matched by a filter F, $|N(F)|$, is proportional to the size of the filter. If the merger of two filters is not larger than the original filters, it will not match more notifications, and it seems therefore reasonable to merge filters where the resulting merger is as small as possible. Put differently, the best merging candidate $F_{\text{mc}} \in \mathscr{F}$ seems to be the filter that has to be "extended the least" to cover G.

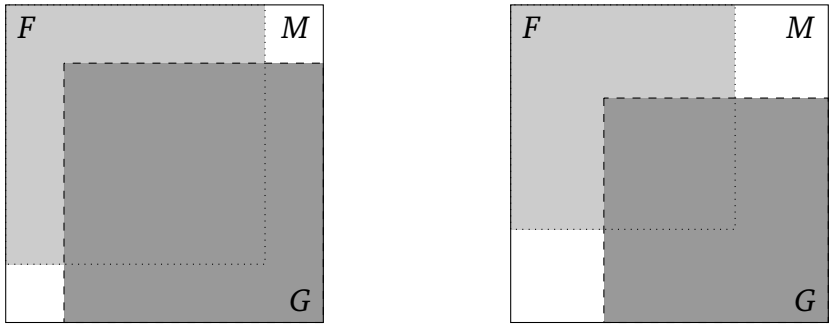
The *size penalty score* s puts the merger size in relation with the sizes of the original filters. We define s for two filters F, G as the size of the merger $F \sqcap G$ divided by the sum of the sizes of F and G.

$$s(F,G) = \frac{|F \sqcup G|}{|F| + |G|} \tag{5.11}$$

The smaller s, the better the merger, and $s = 0.5$ in the best case (when $F \equiv G \equiv M$). This score meets the expectation with respect to the example in Figure 5.8, $s(F_1, G) < s(F_2, G) < s(F_3, G) < s(F_4, G)$.



(a) perfect merger, $s = .75$ **(b)** perfect merger, $s = 1$



(c) imperfect merger, $s \approx .75$ **(d)** imperfect merger, $s \approx 1$

Figure 5.10.: Size penalty scores of pairs of 2-interval filters

Perfect mergers of 1-interval filters always yield a score $0.5 \leq s \leq 1$. If F and G *meet*, $s(F, G) = 1$. This is not the case for N-interval filters (see Figure 5.10), because of the conditions for perfect N-interval filter mergers discussed in Section 5.1.2.

5.3.4 Mean Size-based Merging Penalty

The size penalty score is equally applicable to 1- and N-interval filters, because a general size measure has been defined. For N-interval filters F, G however, we can derive an alternative definition of the size penalty score as the mean of the scores in each filter dimension, i.e., the scores of the (virtual) 1-interval filters F and G are composed of. Let $F = \left(I_{F,i}\right)_{i=1}^{N}, G = \left(I_{G,i}\right)_{i=1}^{N}$. We define the *mean size penalty score* s_{mean} as

$$s_{\text{mean}}(F, G) = \frac{1}{N} \sum_{i=1}^{N} \frac{\left|I_{F,i} \sqcup I_{G,i}\right|}{\left|I_{F,i}\right| + \left|I_{G,i}\right|}. \tag{5.12}$$

This score is not as sensitive to "outlier" dimensions as the regular size penalty score. When the filters F, G have a bad score in some dimension i and a good score in all other dimensions, the penalty is larger with the regular size penalty score, because it impacts the total score in a product, while in the mean penalty score, it contributes in a sum

only. The experimental results presented in Section 5.4 will show that the mean size penalty score variant is superior to the regular size penalty score.

Regarding scores of perfect and imperfect mergers, the only valid statement is that M is a perfect merger if $s_{\text{mean}}(F,G) = 0.5$, which is the case iff $F \equiv G \equiv M$. Since the final result is the arithmetic mean of the scores in the individual dimensions, any score other than 0.5 could potentially result even if the intervals in some dimension would neither intersect nor meet, thus resulting in an imperfect merger.

5.3.5 Distance-based Merging Penalty

Measuring merger quality by filter distance is based on the assumption that filters that are in the proximity of each other in the filter space are likely to match the same notifications. Consequently, the best merging candidate $F_{\text{mc}} \in \mathscr{F}$ for a filter G would be that where $d\left(F_{\text{mc}}, G\right)$ is minimal.

Figure 5.11 shows the distances of the 1-interval filters from the example in Figure 5.8. The intervals D_i are the distance intervals between the filters' centroids such that $d\left(F_i, G\right) = \left|D_i\right|$. Using filter distance as the merging penalty function, $MP{=}d$, the results meet the intuitive expectations, $d\left(F_1, G\right) < d\left(F_2, G\right) < d\left(F_3, G\right) < d\left(F_4, G\right)$.

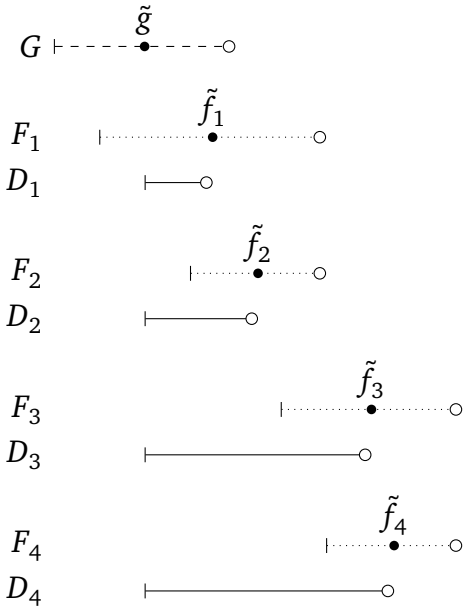
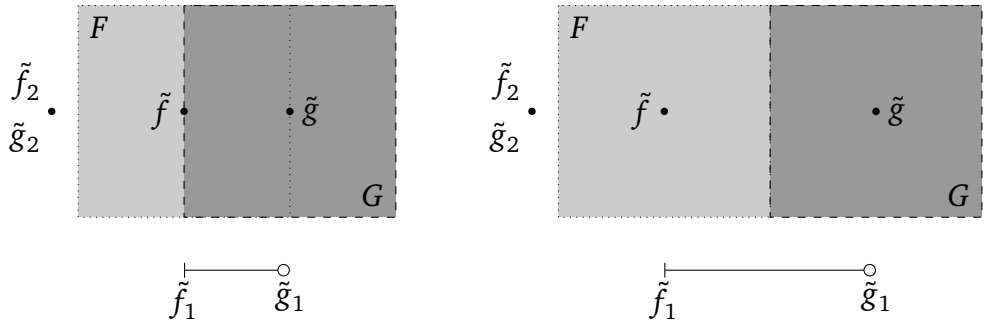


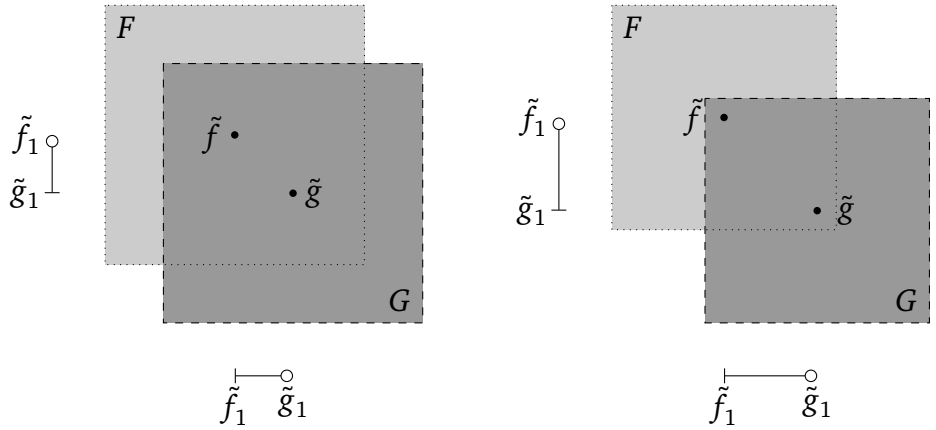
Figure 5.11.: Distance between 1-interval filters

With respect to perfect and imperfect mergers, the only assertion that can be made is that $M = F \sqcup G$ is a perfect merger if $d(F,G) \leq 1$. If $d(F,G) = 2$, M could already be an imperfect merger, namely in the case of $|F| = |G| = 1$. For N-interval filters, the distance value d does not suggest whether perfect or imperfect mergers result, as can be



(a) short distance between medians in dim 1, same median in dim 2

(b) further apart medians in dim 1, same median in dim 2



(c) short distances between medians in both dimensions

(d) longer distance between medians in both dimensions

Figure 5.12.: Distance between 2-interval filters

seen with the examples in Figure 5.12: Since all dimensions are treated equally in the distance sum, the 2-interval filters have pairwise same distances, namely (a) and (c) as well as (b) and (d), although perfect mergers results in (a) and (b) whereas imperfect mergers result in (c) and (d).

5.3.6 Normalized Distance-based Penalty

The penalty score functions for N-interval filters introduced so far exhibit a major drawback: Each filter space dimension is treated equally, while they may pertain to characteristics of the event that are in fact uncomparable. This becomes obvious immediately when realizing that filter space dimensions could be such uncomparable attributes as money and time, quantized in Euros and seconds. Then the distance calculation in Equation (5.7) that adds filter space points would in fact sum up distances in Euros and seconds. The filter space could however be just as well use minutes as chronons and Francs as money quantums, in which case the distance calculation would give a different result.

The important fact is that notification (and filter) distribution can widely vary among the filter space dimensions, i.e., in some filter space dimension, notifications can be much larger and much further apart than in some other, because both metrics are measured in filter space points, while the quantums ("chronons") used to quantize a filter space dimension are independent of each other and could thus lead to dimensions being stretched or compressed relative to the other dimensions. Since filter size and distance are used in the merger quality estimation, the equal treatment of all filter space dimensions disregarding their varying characteristics is a potential source for failure of merger quality estimation.

Therefore, filter space dimensions should be normalized such that the relative stretching or compression is canceled out, resulting in a virtual, normalized filter space.

Normalized Filter Space

Let k_i denote the normalization factor that cancels the relative stretching or compression of a dimension i. The normalized filter space is represented as (isomorph to) $\mathbb{R}^N$. N-interval filters and notifications are represented by polytopes defined by mapping the

respective corner points $\mathbf{p_i^F} \in \mathbb{N}^N$ of the polytopes in the regular filter space onto points $\mathbf{p_i^{F'}} \in \mathbb{R}^N$ according to relation *nfs*:

$$\text{nfs}\colon \mathbb{N}^N \to \mathbb{R}^N, \quad \mathbf{p_i^F} = \begin{pmatrix} p_1 \\ p_2 \\ \dots \\ p_N \end{pmatrix} \mapsto \mathbf{p_i^{F'}} = \begin{pmatrix} k_1 \cdot p_1 \\ k_2 \cdot p_2 \\ \dots \\ k_N \cdot p_N \end{pmatrix} \tag{5.13}$$

The evaluable characteristics of a filter space dimension that give an indication of the relative stretching are the mean distance and mean size of filters and notifications. They can be assumed to be relatively stretched or compressed by the factors k_i. Therefore, we use these characteristics of the notifications to calculate the factors k_i.

Let in the following $\overline{d}_i$ and $\overline{s}_i$ denote the mean distance and size, respectively, of notifications in dimension i, determined by a sample set $\mathcal{N}$.

Let for instance $\overline{d}_1$, $\overline{d}_2$, and $\overline{d}_3$ be the mean notification distances in dimension 1, 2, and 3 of a 3-dimensional filter space, and let $\overline{d}_1 = 4 \cdot \overline{d}_2 = 7 \cdot \overline{d}_3$. Filter space metrics are normalized by stretching dimension 2 by a factor of $k_2 = 4$, dimension 3 by a factor of $k_3 = 7$, while leaving dimension 1 as it is, $k_1 = 1$.

In general, the distance-based normalization factor k_i^{d} of dimension i in an N-dimensional filter space is the relation of the maximum mean distance to the mean distance in dimension i:

$$k_i^{\text{d}} = \frac{\overline{d}_{\max}}{\overline{d}_i}, \quad \overline{d}_{\max} = \max\left(\overline{d}_1, \overline{d}_2, \dots, \overline{d}_N\right)$$

Accordingly, the size-based normalization factors are given by:

$$k_i^{\text{size}} = \frac{\overline{s}_{\max}}{\overline{s}_i}, \quad \overline{s}_{\max} = \max\left(\overline{s}_1, \overline{s}_2, \dots, \overline{s}_N\right)$$

Both mean distance and mean size are merely indications of the relative stretching or compression of dimensions, and could obviously result in different factors k_i. To exploit both characteristics for a calculation of the normalization factors, the factors can simply be added[2], which gives the integrated normalization factors

$$k_i = \frac{\overline{d}_{\max}}{\overline{d}_i} + \frac{\overline{s}_{\max}}{\overline{s}_i}. \tag{5.14}$$

[2] The correctness becomes obvious considering that the arithmetic mean $\frac{1}{2}\left(k_i^{\text{dist}} + k_i^{\text{size}}\right)$ could be used here. The factors are however relative by nature, and the division by 2 does not change the relation.

Inserting in Equation (5.13) the factors k_i thus calculated, notifications of the sample set $\mathcal{N}$ exhibit approximately the same mean size and distance in all dimensions of the normalized filter space. The filter space is thus normalized with respect to $\mathcal{N}$. The better the relative per-dimension characteristics of the sample notification set approximate the relative per-dimension characteristics of all notifications, the better a normalization is achieved with respect to the overall notification distribution.

Enhancing the Distance-based penalty score

The normalized virtual filter space can be used to improve the distance-based penalty score.[3] Instead of calculating the distance between the centroids of the original intervals, distances between the centroids of the polytopes representing the filters in the normalized filter space are calculated. The centroid $\tilde{\mathbf{f}}'$ of the polytope with corner points $\mathbf{p_1^{F'}}, \mathbf{p_2^{F'}}, \ldots, \mathbf{p_N^{F'}}$ representing the N-interval filter F in the normalized filter space is given by

$$\tilde{\mathbf{f}}' = \frac{\mathbf{p_1^{F'}} + \mathbf{p_2^{F'}} + \cdots + \mathbf{p_N^{F'}}}{2^N}$$

We define the *normalized filter distance* d_{norm} as L_1 distance between these centroids.

Definition Normalized Filter Distance: *The normalized distance d_{norm} of two N-interval filters $F = \left(I_{F,i}\right)_{i=1}^{N}, G = \left(I_{G,i}\right)_{i=1}^{N}$ is the L_1 distance between the centroids $\tilde{\mathbf{f}}'$ and $\tilde{\mathbf{g}}'$ of the polytopes that represent F and G in the virtual filter space.*

$$d_{\text{norm}}(F,G) = d_1\left(\tilde{\mathbf{f}}', \tilde{\mathbf{g}}'\right) = \left|\tilde{\mathbf{f}}' - \tilde{\mathbf{g}}'\right|_1 = \sum_{i=1}^{N} \left|\tilde{f}_i' - \tilde{g}_i'\right| \tag{5.15}$$

Since the normalized filter space is isomorph to $\mathbb{R}^N$, Euclidean distance and L_1 distance can be used equally well. However, it does not affect the results whether one or the other is applied, because filter distances are used only in relative comparisons, and this relation is the same for Euclidean and L_1 distance. For instance, if $d(F,G_1) < d(F,G_2)$ using L_1 distance, the same holds with Euclidean distance calculation.

The calculation of the normalization factors k_i based on mean notification distance and mean notification size however requires prior evaluation of a sample notification set

[3] The size-based penalty scores could likely also benefit, but we concentrate on the distance-based penalty score, because it outperforms the size-based one as detailed later when presenting experimental results in Section 5.4 for reasons discussed there.

$\mathcal{N}$. The general approach can also be applied with an online analysis of notifications, which would promise better results in the case of dynamically changing notification characteristics. This would however preclude the calculation and storage of the normalized filter centroid upon filter creation, which significantly increases efficiency. We will therefore assume that a suitable sample set of notifications is available and the normalization factors k_i are calculated once and remain fix throughout system runtime.

5.4 Experimental Results

Two general approaches of evaluating the quality of a merger in a merging penalty function *MP* have been presented in the previous sections, the filter size penalty score s and the filter distance d, both with variants. Each has been defined for 1- and N-interval filters. These alternatives for *MP* can be employed in the *findMergingCandidate* function to find a merging candidate for a filter G from a set $\mathcal{F}$, which can in turn be applied to heuristically solve N1FMP.

For the experiments, the heuristical algorithm for N1FMP was extended to represent in a more realistic way the filter handling of a broker in the notification service by additionally checking for covering relations among filters. The experimental setup is described in Subsection 5.4.1. With respect to the applied merging strategies, the exhaustive *vs.* non-exhaustive approach to merging candidate search is compared on one hand (Subsection 5.4.2), and the alternative penalty score functions are evaluated and compared on the other hand. Results for the size-based penalty score functions are discussed in Subsection 5.4.3, and for the distance-based ones in Subsection 5.4.4. Subsection 5.4.5 finally discusses the impact of filter and notification characteristics and their distribution in the filter space, and the effect of varying sizes of the set of filters to be merged.

5.4.1 Experimental Setup

The overall approach taken here to assess a merging strategy is as follows. Given a set of filters $\mathcal{G}$, an intially empty set of filters $\mathcal{F}$ is populated by adding or merging each $G \in \mathcal{G}$ to $\mathcal{F}$, one after the other. In this process it is first checked for every filter G if a cover relation exists between G and any of the filters in $\mathcal{F}$, in which case the covered filter is dropped. If G has not been dropped, $\mathcal{F}$ is searched for a merging candidate using one of the *findMergingCandidate* alternatives. If a merging candidate F_{mc} is found, the merger $M = F_{\text{mc}} \sqcup G$ is created. Finally, it is checked if this merger covers any $F \in \mathcal{F}$, which is dropped in this case. Figure 5.13 presents the implementation of the experimental setup in pseudocode.

Input: Original set of filters $\mathscr{G}$
Output: Processed set of filters $\mathscr{F}$

$\mathscr{F} \leftarrow \emptyset$
foreach $G \in \mathscr{G}$ **do**
    foreach $F \in \mathscr{F}$ **do**
        if $F \sqsupseteq G$ **then**
            `// continue for-loop with next` G
            continue G
        end
        if $G \sqsupseteq F$ **then**
            $\mathscr{F} \leftarrow \mathscr{F} \setminus F$
        end
    end
    $F_{\text{mc}} \leftarrow$ `findMergingCandidate(`$\mathscr{F}$`,` G`);`
    if $F_{\text{mc}} = null$ **then**
        $\mathscr{F} \leftarrow \mathscr{F} \cup \{G\}$
    else
        $M \leftarrow F_{\text{mc}} \sqcup G$
        $\mathscr{F} \leftarrow \mathscr{F} \cup \{M\} \setminus \{F_{\text{mc}}\}$
        foreach $F \in \mathscr{F}$ **do**
            if $M \sqsupseteq F$ **then**
                $\mathscr{F} \leftarrow \mathscr{F} \setminus F$
            end
        end
    end
end
return $\mathscr{F}$

Figure 5.13.: Pseudocode: Experimental algorithm – Adding/merging each $G \in \mathscr{G}$ sequentially into $\mathscr{F}$ applying some merging strategy.

After this run, the achieved reduction $r(\mathscr{F})$ and resulting precision $r_{\mathscr{N}}(\mathscr{F})$ are calculated using a large sample set of notifications $\mathscr{N}$.

For each test, the algorithm was performed for a large number of different penalty score threshold values p_0, each run resulting in a precision value p and a reduction value r. The comparison of these sets of values derived for different merging strategies allows comparing the effectiveness of the strategies.

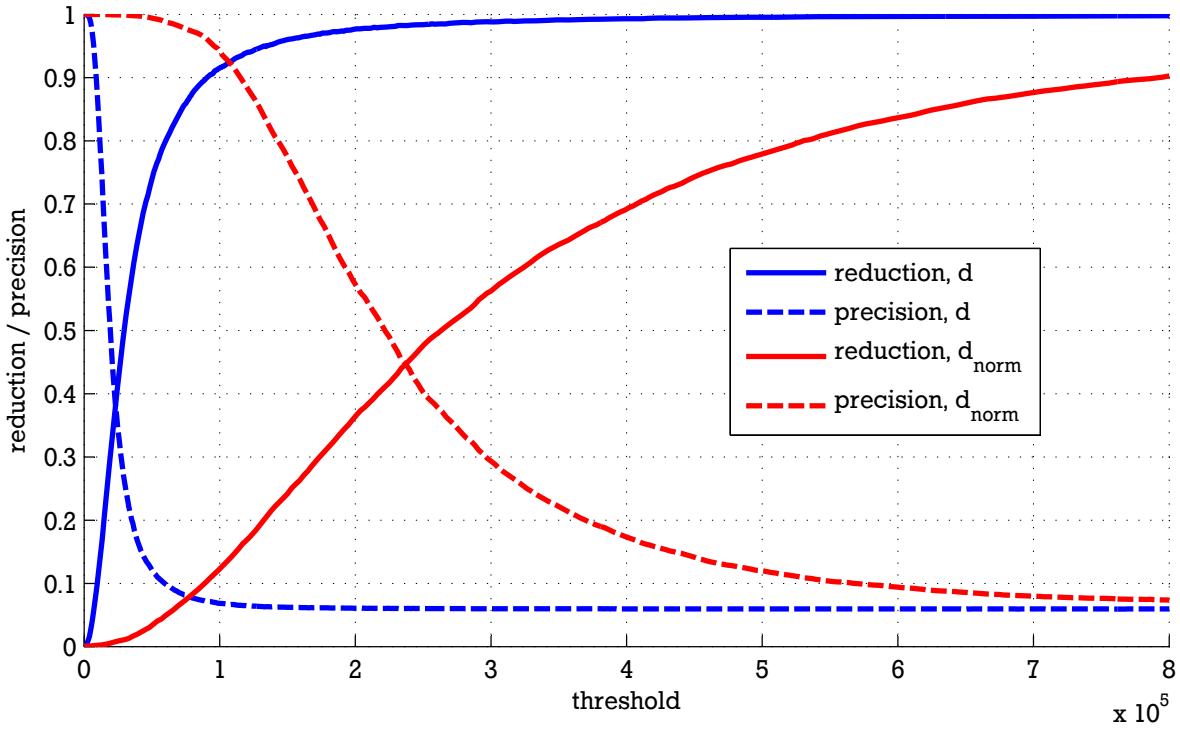
The results for the different merging approaches and test data sets are in the following compared by means of plots of precision and reduction. As an introductory example, Figure 5.14 shows different representations of the experimental results comparing the distance penalty score and the normalized distance penalty score. The data for the plots was generated by executing the test algorithm with a set of 10,000 2-interval filters $\mathscr{G}$ and 500 different penalty score threshold values $0 \leq p_0 < 1000$, once using the distance penalty score d as *MP*, and once using the normalized distance penalty score d_{mean} as *MP*. For each of the resulting 2×500 sets $\mathscr{F}$, the reduction was calculated and, using a set of 100,000 2-interval notifications $\mathscr{N}$, the matching precision was estimated.[4] The reduction and precision values are plotted over the threshold values in Figure 5.14a. The discrete (p_0, r) and (p_0, p) pairs have been linked to continuous curves for presentation purposes.

Generally, low penalty score threshold values result in few filter mergers, and consequentially, low reduction and high precision. An increasing threshold leads to the creation of more mergers, and the reduction increases and the precision decreases.

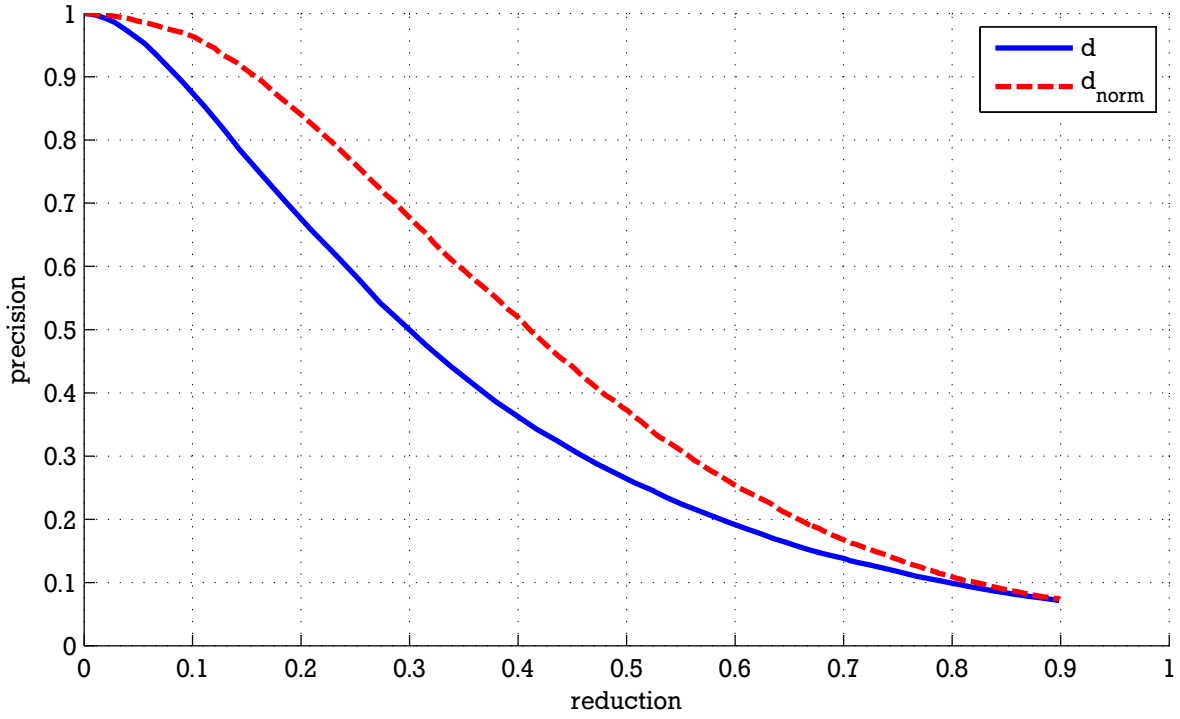
Obviously, the plots of reduction and precision over threshold do not serve well to compare the results of two different strategies. It is hard to read from these plots, which one of the strategies performed better, i.e., achieved higher filtering quality. This is because a threshold value results in different reductions with different strategies, and precision depends on reduction, not on threshold value. Hence, we want to compare the achieved precision at a given reduction irregardless of the threshold value that was used. The plots of precision over reduction as in Figure 5.14b serve for that. The plots were created by connecting the 500 (r, p) value pairs for each strategy to form continuous curves. This representation has the advantage that the achieved filtering quality can easily be compared for different penalty score functions, while widely differing threshold value ranges were used. We will in the following exclusively use such plots of precision over reduction to present and compare results of different penalty score functions. All plots were generated as described for this example.

The assessment of the merging strategies was performed using sample sets of synthetic, randomly generated (2-)interval filters and notifications. These sets were created

[4] Filter set $\mathscr{G}$ and notification set $\mathscr{N}$ were created from sample data sets $\mathscr{R}_{11}$ and $\mathscr{R}_{14}$, respectively. See Appendix B.



(a) Reduction and precision over penalty score threshold



(b) Precision over reduction

Figure 5.14.: Results for penalty score function alternatives d (regular) and d_{norm} (normalized)

from a pool of 11 sets of 110,000 integer intervals $\mathscr{I}_1, \ldots, \mathscr{I}_{11}$ and 11 sets of 110,000 2-intervals, i.e., records of two intervals interpreted as intervals in different filter space dimensions taking the form of rectangles in the 2-dimensional filter space, $\mathscr{R}_1, \ldots, \mathscr{R}_6$ and $\mathscr{R}_{11}, \ldots, \mathscr{R}_{17}$. The composition and exact characteristics of the synthetic sample data sets are detailed in Appendix B.

The sample sets differ widely in the distribution of the intervals in the filter space and of their sizes. In some sets, the intervals are uniformly distributed in the filter space, whereas in others, they are clustered around a number of hotspots in a normal distribution with the hotspot value as μ and varying standard deviations σ.[5] The sizes of the intervals in all sets follow a half-normal distribution (only the positive values in a normal distribution with $\mu = 0$) shifted by $+1$ to have only sizes ≥ 1) with different standard deviations $\sigma_{|I|}$. The impact of the characteristics of filters and notifications is discussed later in Subsection 5.4.5.

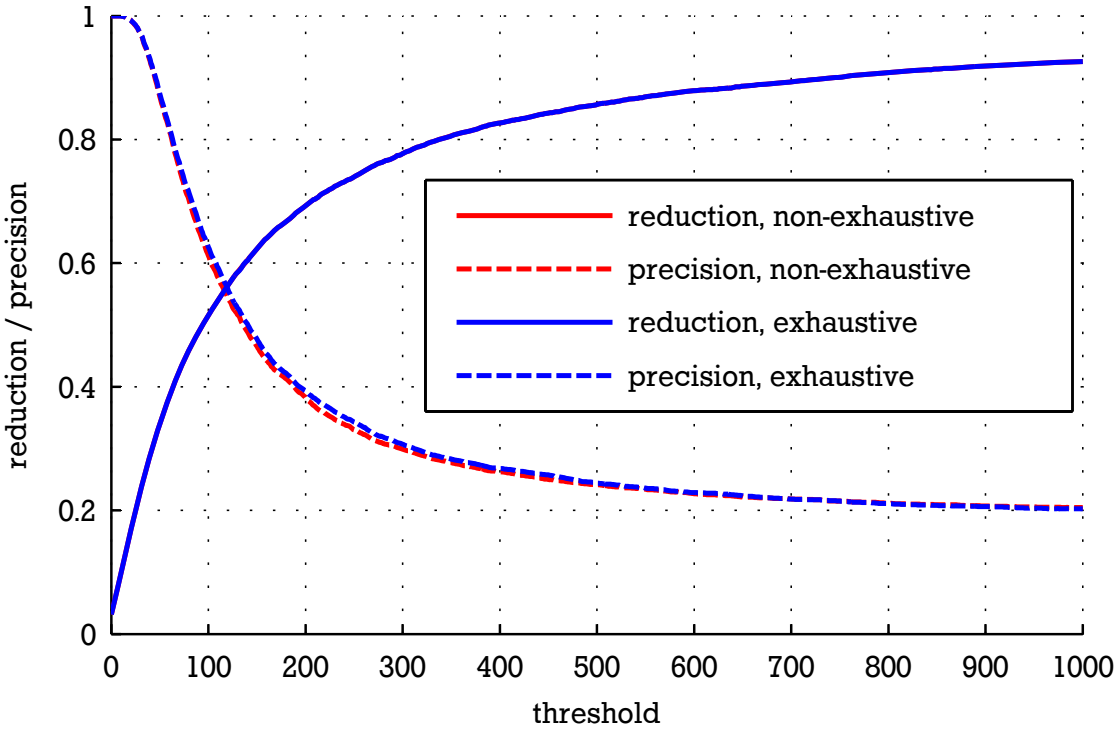
5.4.2 Merging Candidate Search Alternatives

Figure 5.15 shows results comparing the exhaustive and non-exhaustive implementations of *findMergingCandidate*. The data for the plots was generated as described above[6] using the distance penalty score function, once using the exhaustive candidate search approach, and once using the non-exhaustive one.

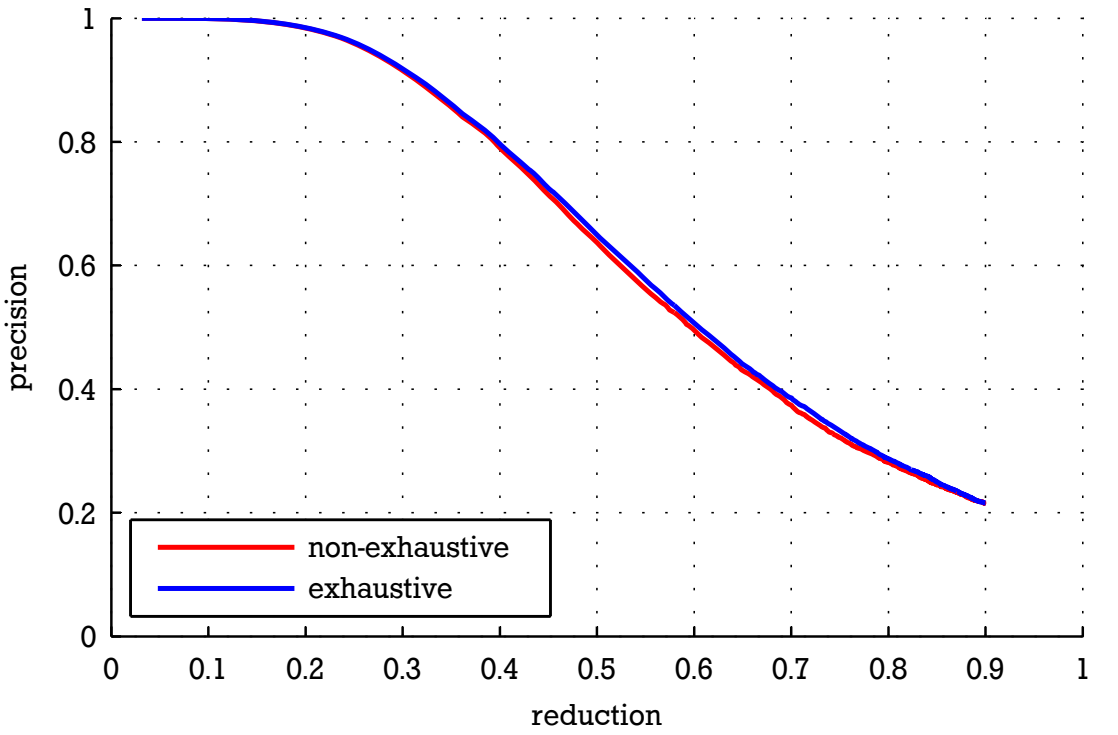
The number of created mergers for a given penalty score threshold is the same for both, resulting in the same reduction plot. The filtering quality of the filter set $\mathscr{F}$ however differs, resulting in slightly higher precision values for the exhaustive approach. The exhaustive approach to merging candidate selection overrules the non-exhaustive approach with respect to achieved precision, as expected. While the experiments showed that this result holds irregardless of the applied penalty score function and the characeristics of the sample data sets, the achieved precision gain is however negligible. This is probably due to two reasons: Firstly, in most of the successful runs of *findMergingCandidateNonExhaustive* (i.e., when a candidate is found), there would have been very few other filters (if any) qualifying as merging candidates. Hence, chances are that the returned candidate has a merging penalty value close to the one that *findMergingCandidateExhaustive* would have found, or it even is the same filter. This assumption is supported by the same reduction plots. Secondly, determining a penalty score to estimate the quality of a merger is also a heuristical approach, where the best overall

[5] Clusters in the distribution of the data respresent "interest hotspots" when used as filters and "event hotspots" when used as notifications. The existence of such hotspots can be assumed for many real-world applications.

[6] A set of 10,000 1-interval filters $\mathscr{G}$ created from set $\mathscr{I}_2$ and a set of 100,000 1-interval notifications $\mathscr{N}$ created from set $\mathscr{I}_1$ was used as test data.



(a) Reduction and precision over penalty score threshold



(b) Precision over reduction

Figure 5.15.: Results for the exhaustive and non-exhaustive approach to merging candidate search

merging candidate does not necessarily have the lowest *MP* function value. This further limits the impact of the choice of the alternative for *findMergingCandidate*.

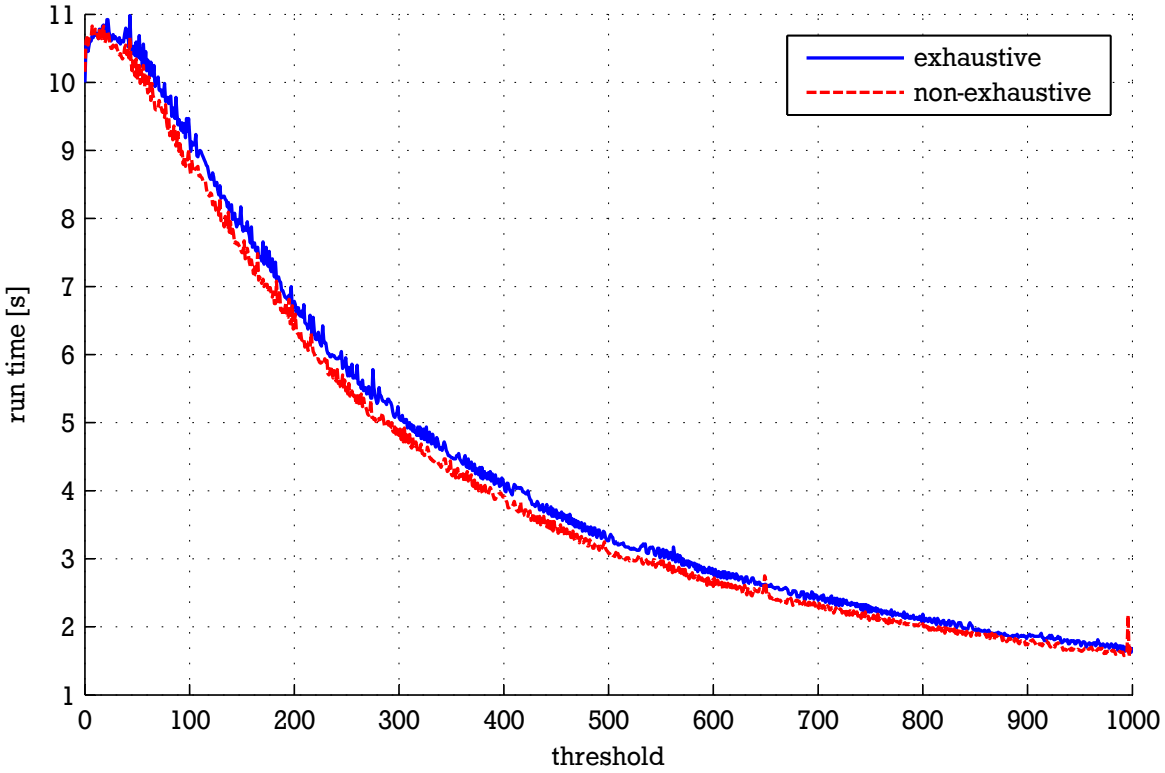


Figure 5.16.: Run-times of the exhaustive and non-exhaustive approach to merging candidate search

Regarding the performance however, it turned out that the achievable benefit of the non-exhaustive approach is negligible as well. The algorithm runs slightly faster in most cases for the non-exhaustive approach, but the example plots in Figure 5.16 show how small the difference is. It shows the run times of the runs of the experiment algorithm that produced the results plotted in Figure 5.15.

The reason for the small difference is that in most cases, *findMergingCandidate* is not successful. In these cases the non-exhaustive approach's run-time performance also degrades to $O(|\mathscr{F}|)$. In addition, first checking for covering relations requires searching $\mathscr{F}$ exhaustively anyways. In the case of a successful merging candidate search, it is even searched a second time to find covered filters. This crucially limits the achievable overall performance gain of the non-exhaustive approach.

The finding that the exhaustive and non-exhaustive approaches to merging candidate selection exhibit negligibly small differences with respect to the achieved precision as well as to the run-time performance holds for 1- and N-interval filters, using different

penalty score functions and different sets $\mathscr{G}$ and $\mathscr{N}$. In the following discussions, it is therefore not distinguished anymore between these two approaches.

5.4.3 Size-based Penalty Score Functions

The most important finding regarding the size-based penalty score functions is that it is unusable for larger reductions, usually above approximately 20–40 %. At this point the overall approach degrades thus that merger clusters emerge: Very large mergers are created that are predominantly selected as merging candidates and eventually only very few huge mergers remain in the destination set $\mathscr{F}$. Figure 5.17 shows typical reduction plots for different source filter sets $\mathscr{G}$ of 2-interval filters with varying sizes and distribution patterns. At threshold value p_{degrad}, where the effect appears (marked with larger dots in the plot), the reduction boosts to ≈ 1, because less than ten huge mergers of the originally 10,0000 filter remain. The behavior can be observed for both the regular size penalty score s and its variant, the mean size penalty score s_{mean}.

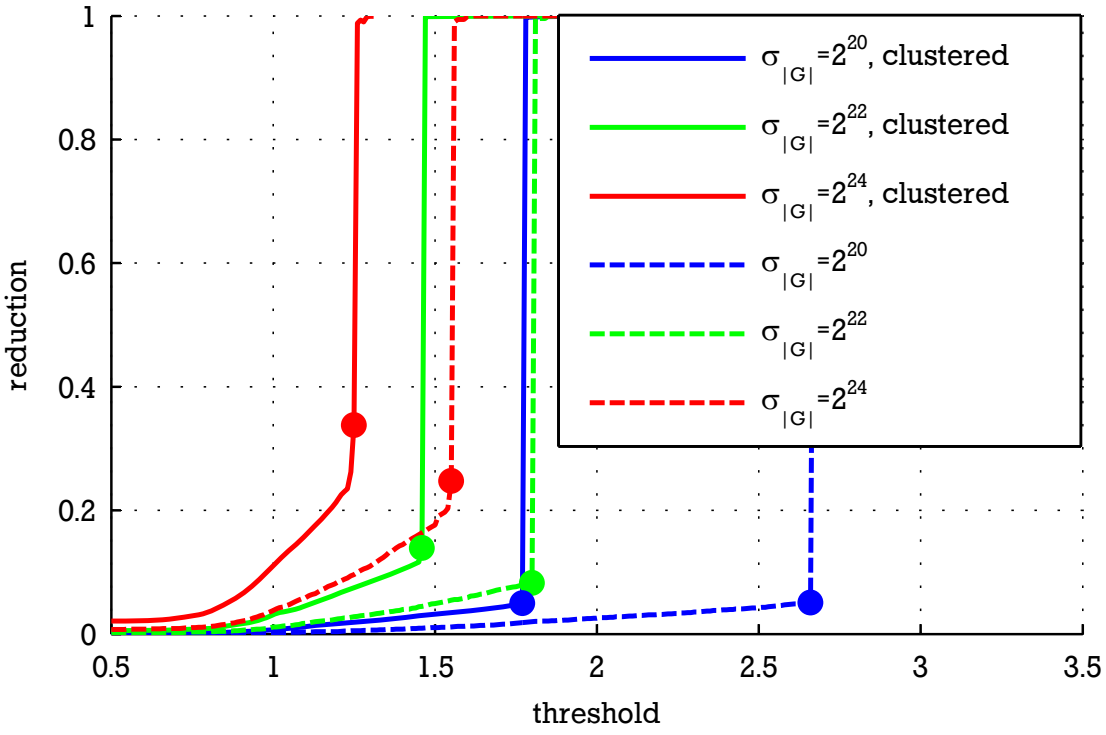
The reason for this behavior is that the size-based penalty scores s and s_{mean} use filter size in a relation of merger and original filters (Equation (5.11)). The larger a filter $F \in \mathscr{F}$ the more likely F is selected as merging candidate, because the merger's relative size $|F \sqcup G|/|F|$ is lower than for small filters, which leads to a better (lower) size penalty score.

The example in Figure 5.18 shows such a disadvantageous comparison of two filters F_1, F_2 as merging candidates for G. The penalty score definition favors those filters as merging candidates that have to be "extended the least" to include G. In the case of the smaller filter, F_1 has to be "extended" to four times its size to become M_1 and to include G. In the case of the larger filter, M_2 is not even twice as large as F_2. Hence, M_2 is created, and for the next filter G, chances are even higher that M_2 is chosen again. The effect increases and merger clusters emerge.

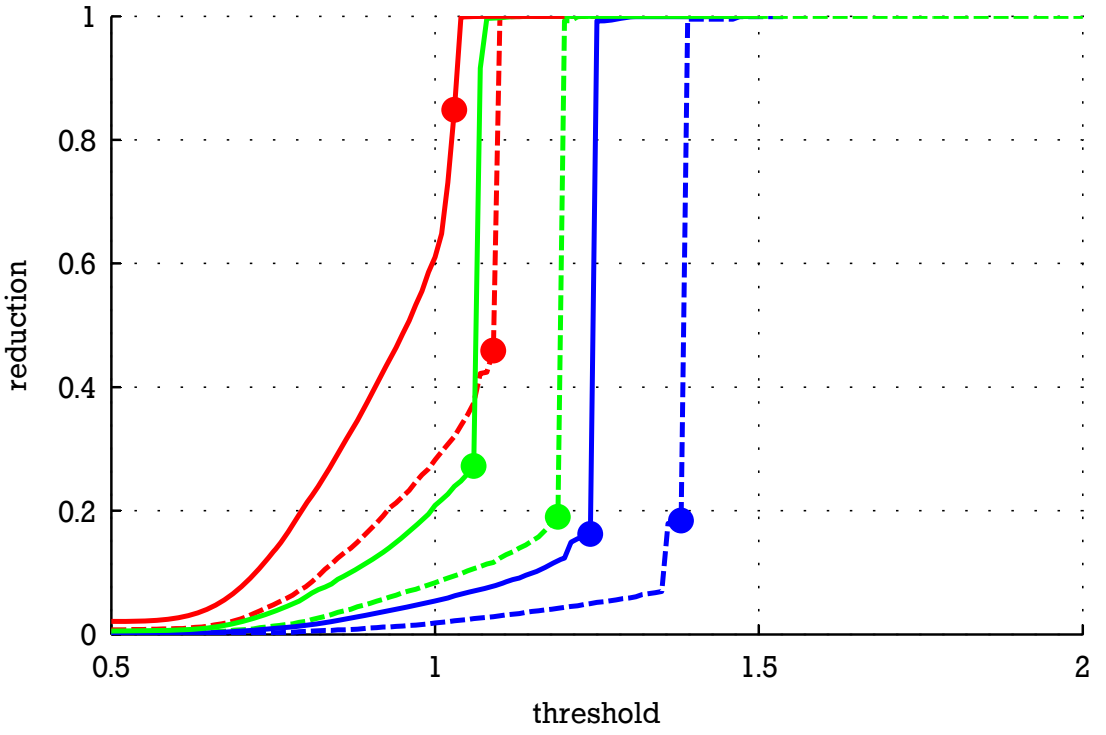
The value of p_{degrad} depends on the density of the filter set $\mathscr{G}$ in the filter space. The six different filter sets used for the plots in Figure 5.17 were created from 10,000 2-interval filters each, distributed in the value range $[0, 2^{20}] \times [0, 2^{20}]$. In three of the sets, the filter distribution is uniform in the filter space, in the other three it is clustered (in a normal distribution with $\sigma = 2^{16}$ in each dimension around ten hotspots). Obviously, the higher $dens(\mathscr{F})$ the lower p_{degrad}. Since the clustered filter sets exhibit higher local densities than the ones with uniform distribution, and sets with larger filters exhibit higher densities in general, p_{degrad} is lower with those. However, larger reductions can be achieved with those more dense sets[7] before clusters emerge.[8] The same observation

[7] For the filter sets used for the plots in Figure 5.17, the achieved reduction at p_{degrad} was $\approx 5\,\%$, $8\,\%$, and $25\,\%$ for the unclustered, and $5\,\%$, $14\,\%$, and $34\,\%$ for the clustered filter sets.

[8] The reason for this is discussed below when examining the impact of filter set characteristics.



(a) Size penalty score s



(b) Mean size penalty score s_{mean}

Figure 5.17.: Degradation of the size-based penalty scores. Above some threshold value p_{degrad}, the reduction boosts to ≈ 1.

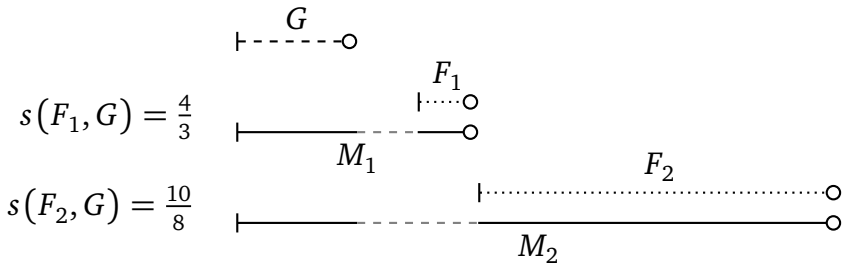


Figure 5.18.: Example for emerging merger clusters caused by large filters being favored as merging candidates in the size-based penalty scores.

can be made comparing s and s_{mean}. For the plots in Figure 5.17, the same filter sets were used for both penalty scores. The difference is that p_{degrad} is generally lower with s_{mean} but larger reduction is achieved.[9]

In the "stable" merging threshold and reduction range, the size-based penalty score functions however provide good results. Figure 5.19 shows examples for clustered and unclustered sets of 2-interval filters and notifications.[10] The plots have been truncated where the precision drops below 10 %. Reduction values of 5–20 % can be achieved using the size-based penalty score function without considerably reducing precision. The precision plots exhibit low negative slope at low reduction values, which however increases at higher values. While s produces slightly better results at very low reduction where the negative slope of the precision plot is low, this slope does not increase as much for the variant s_{mean} resulting in higher precision at high reduction than with s. The s_{mean} approach also seems more stable with respect to the degradation effect, which appears at much higher reduction.

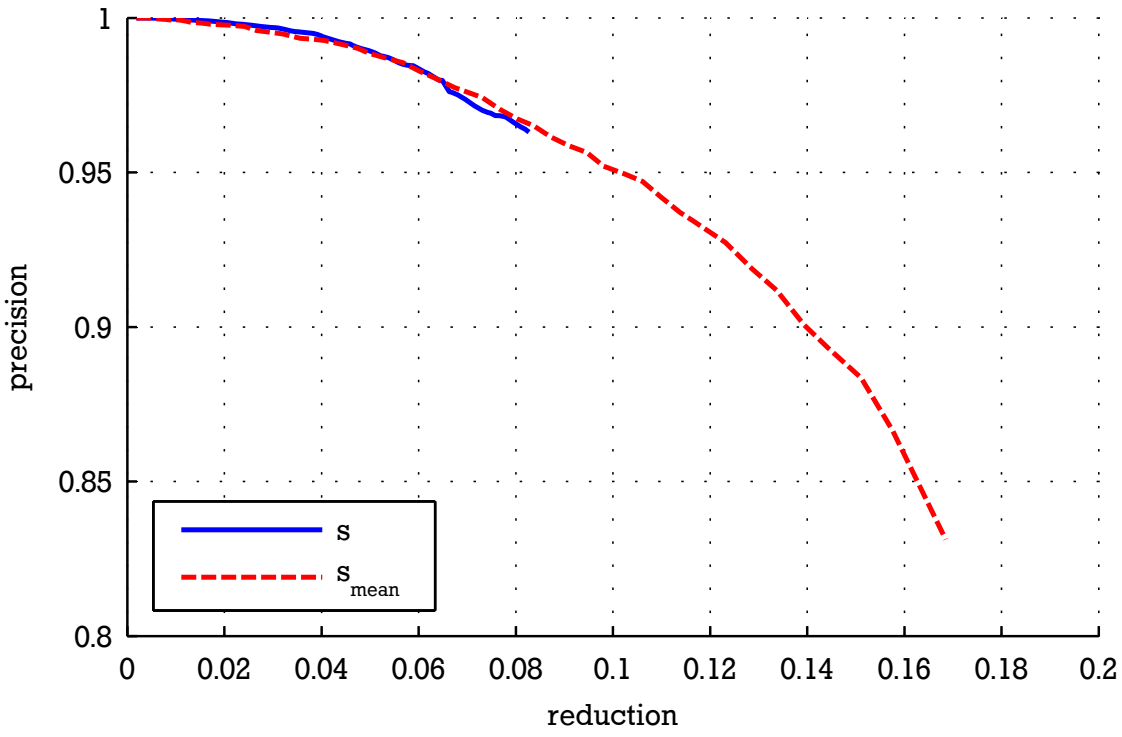
5.4.4 Distance-based Penalty Score Functions

The distance-based penalty score function d does not exhibit a degradation behavior similar to the size-based one. It produces stable results for all penalty score threshold values. This is because it is based solely on one filter characteristic, the centroid, which is not changed by a merge operation in such a way that the filter is more likely to produce a lower penalty score in the future.

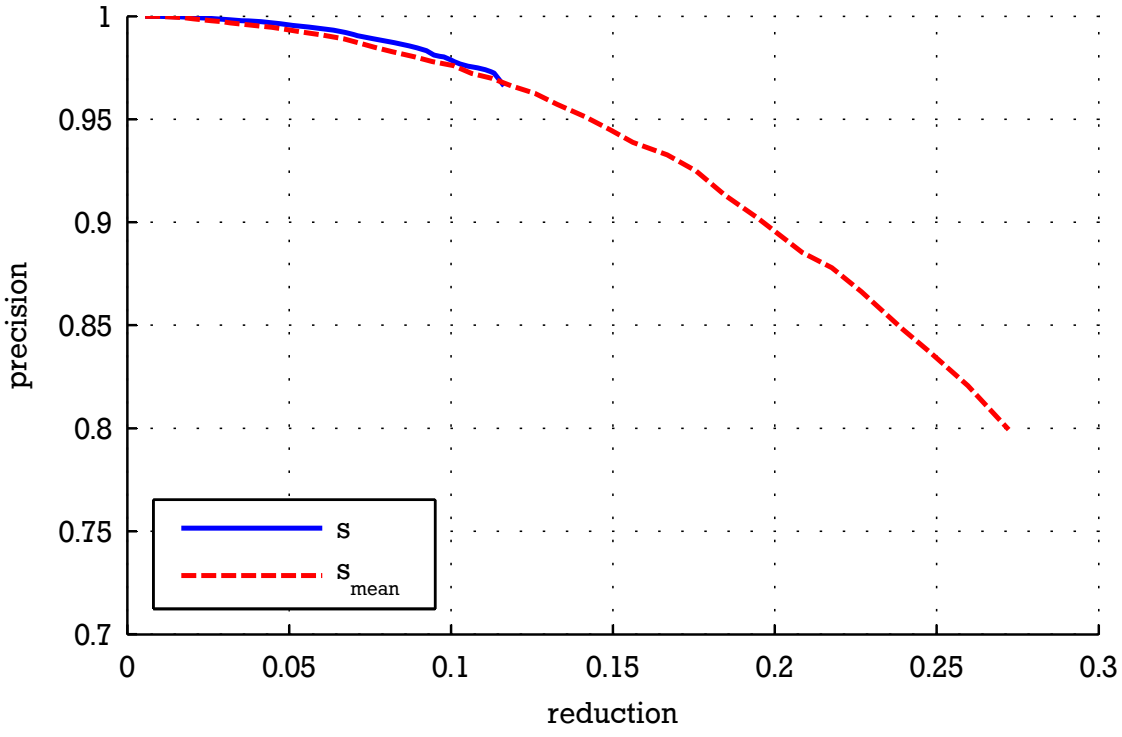
In a direct comparison with the size penalty scores, the distance penalty score generally achieves higher filtering quality for any sets of filters and notifications. Figure 5.20 shows the results for the same sets of 2-interval filters and notifications in a direct comparison of the distance penalty score and the mean size penalty score. The experimental

[9] The reduction for threshold value p_{degrad} was approximately 18 %, 19 %, and 46 % for the unclustered, and 16 %, 27 %, and 85 % for the clustered filter sets.

[10] Filter and notification sets were generated from sets $\mathscr{R}_2$ (uniformly distributed) and $\mathscr{R}_5$ (clustered).

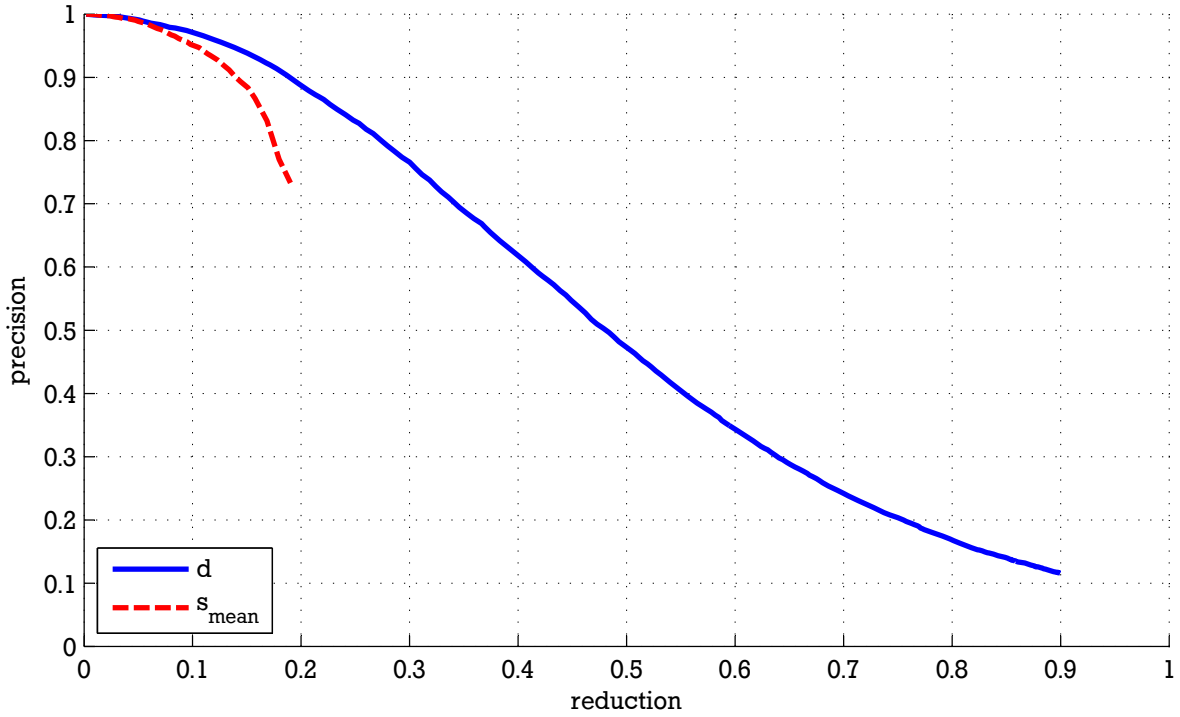


(a) Uniformly distributed

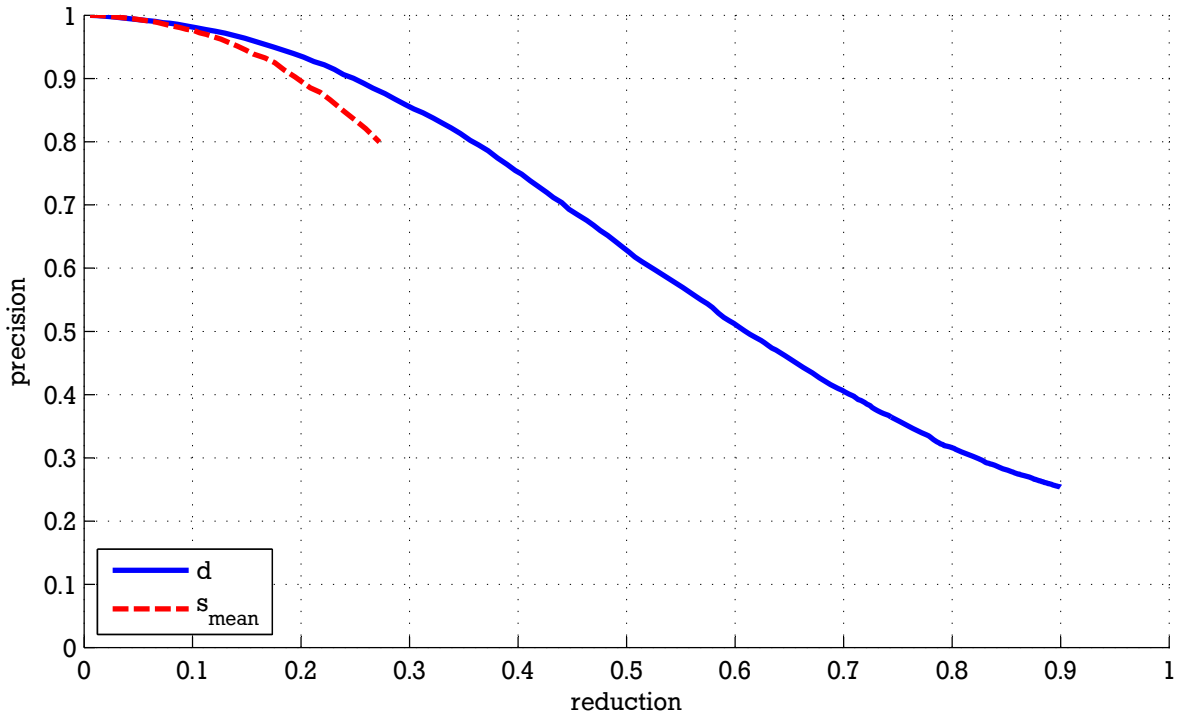


(b) Clustered

Figure 5.19.: Results for the size-based penalty score functions s and s_{mean} for sets of clustered and unclustered 2-interval filters and notifications



(a) Uniformly distributed



(b) Clustered

Figure 5.20.: Comparison of the distance penalty score and the mean size penalty score for sets of uniformly distributed and clustered 2-interval filters and notifications

results for all data sets show that the distance penalty score generally outperforms the size-based penalty scores with respect to filtering quality.

The sets of 2-interval filters and notifications used to produce the results presented so far all exhibited the same characteristics in both dimensions, i.e., the mean distance and mean size of filters and notifications are equal in both dimensions. With those data sets, the normalized distance penalty score d_{norm} produces the same results as the regular distance penalty score d. In contrast, Figure 5.21 shows the results for the two penalty score functions using sets of 2-interval filters $\mathscr{G}$ and notifications $\mathscr{N}$ that exhibit different characteristics in the two dimensions.[11] The normalized distance penalty score achieves a higher filtering quality, as expected.

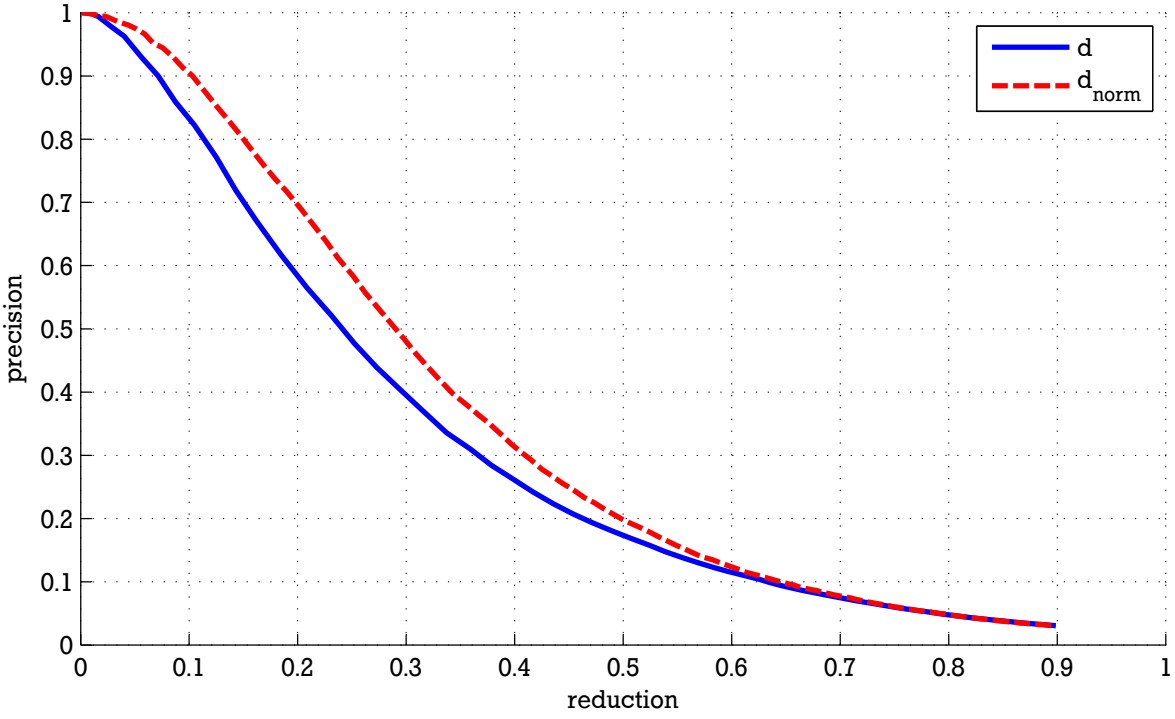


Figure 5.21.: Comparison of the regular distance penalty score d with the normalized distance penalty score d_{norm} for sets of 2-interval filters and notifications with different dimension characteristics

The intervals in these sets are uniformly distributed in the range $[0, 2^{23}]$ in dimension 1, and in the range $[0, 2^{20}]$ in dimension 2; the interval size follows a half-normal distribution with $\sigma_{|F|} = \sigma_{|n|} = 2^{13}$ in dimension 1, and $\sigma_{|F|} = \sigma_{|n|} = 2^{10}$ in dimension 2.[12] Hence, dimension 1 is stretched approximately by factor 8 compared to dimension

[11] Both sets were created from 2-interval set $\mathscr{R}_{11}$.

[12] Here and in the following discussions, we denote with $\sigma_{|G|}$ the standard deviation of the filter size $|G|$. The standard deviation of the notification size $|n|$ is denoted with $\sigma_{|n|}$.

2. In this case, dimensions of the normalized filter space are stretched by $k_1 = 2$ and $k_2 \approx 16$ (according to Equations (5.14) and (5.13)).

The normalization approach based on notification distribution characteristics also succeeds in all cases where notifications are distributed differently than filters, i.e., with larger or smaller mean size or mean distance in any dimension. Figure 5.22 shows two examples, where the same filter set was used as before but notification sets with (a) larger mean distance in dimension 1 and (b) smaller mean size in dimension 2.

Since the distance-based penalty scores have turned out to be better than the size-based ones in all aspects, we used for subsequent tests only the (normalized) size-based penalty score.

5.4.5 Impact of Sample Data Characteristics

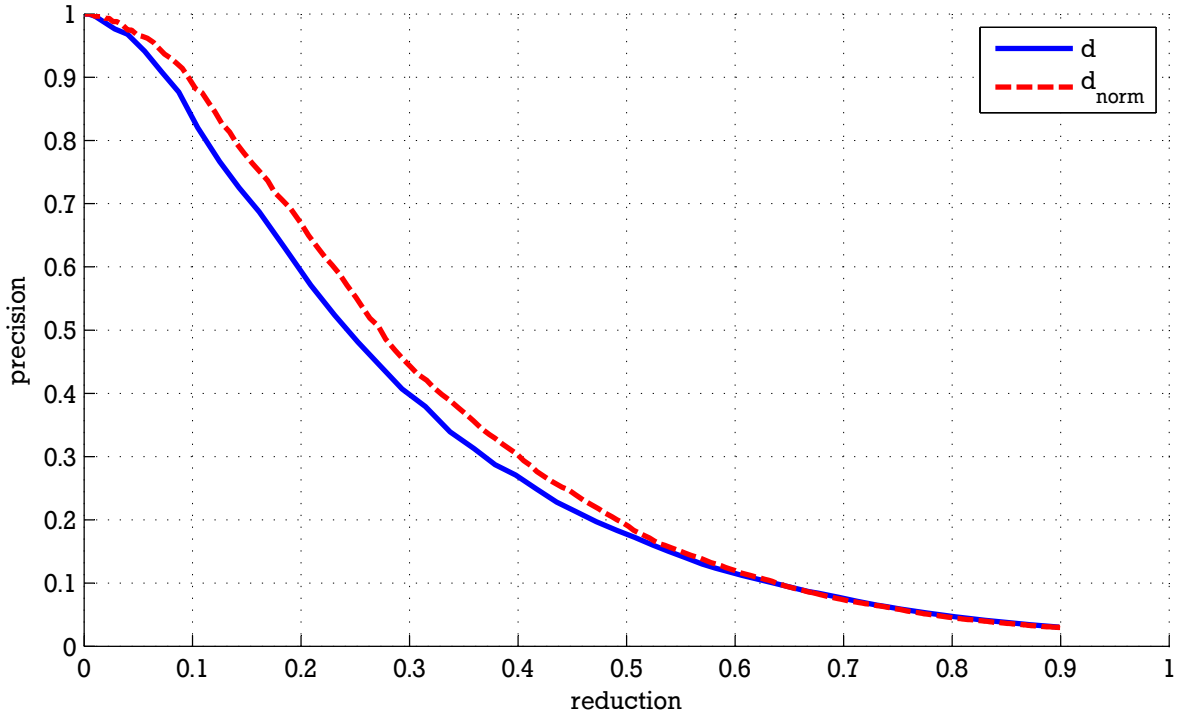
Using synthetic sample data sets with varying characteristics allowed to test the strategies under different conditions and derive general statements about the effects of the characteristics on the performance of the merging strategies.

Filter and Notification Characteristics

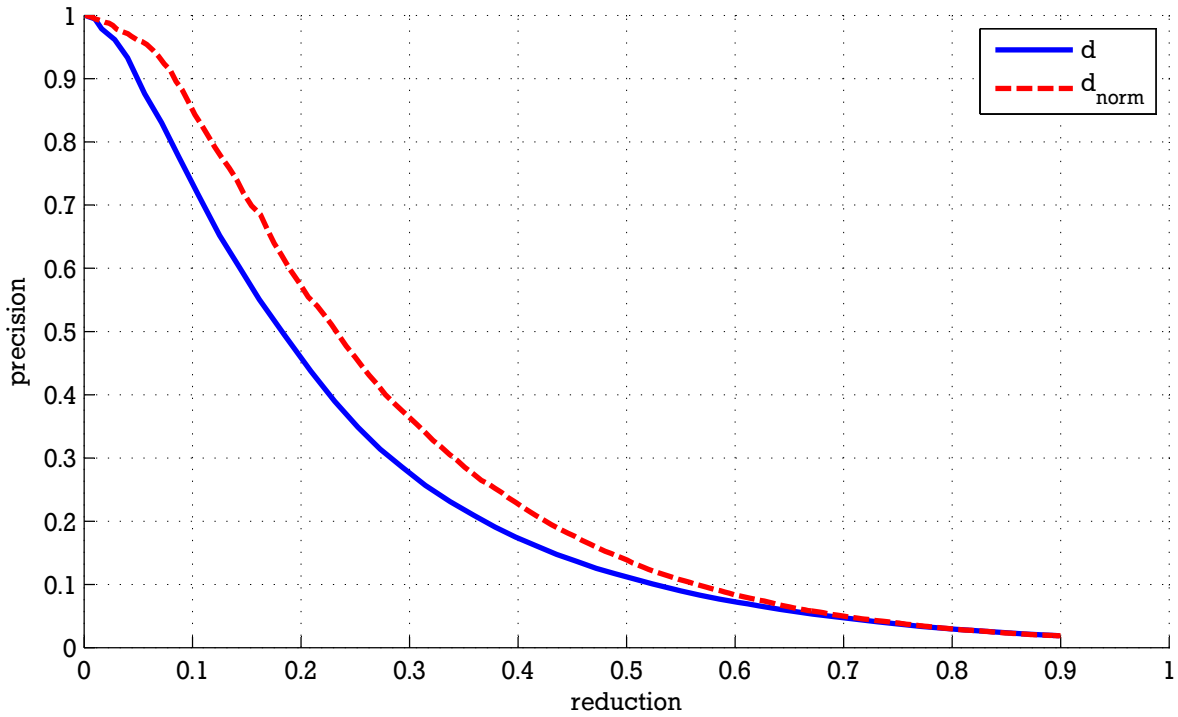
The example plots in Figure 5.23 show results for different sets of 1-interval filters and notifications, which vary in the size of the intervals and the distribution in the filter space. The intervals in all sample data sets are distributed in the range $\left[0, 2^{20}\right]$, uniformly in (a) and clustered in (b).

We see that filtering quality is reduced less significantly the higher the filter and notification density in the filter space. With a fixed amount of filters in a bounded filter space, the density is proportional to the filter size, and the probability that a notification n is matched by any $F \in \mathscr{F}$ is proportional to the density. The higher this probability the less likely it is that this notification becomes a false positive through merging. Large notifications are more likely to be matched and large filters are more likely to match.

In the examples here, the filter space is bounded; its size is $|\mathbb{F}| = 2^{20}$. Since the same amount of filters is used in each run, the filter density is higher for sets with larger filters. With half-normally distributed filter sizes with $\sigma_{|G|} = 8$, the mean filter size is $\overline{|G|} \approx 7.4$, and according to Equation (5.9), the filter density with respect to $\mathscr{G}$ is $dens(\mathscr{G}) \approx 0.07$. With four times the standard deviation, $\sigma_{|G|} = 32$, the mean size increases to $\overline{|G|} \approx 26.5$, and the density to $dens(\mathscr{G}) \approx 0.25$. The densities in the clustered data sets are much higher locally around the hotspots, and thus higher precision can be achieved at a given reduction value compared to the uniformly distributed data sets. The relative effect is nevertheless the same. With high densities (filters and/or notifications), large reductions can be achieved without significantly impacting the precision. In the examples in

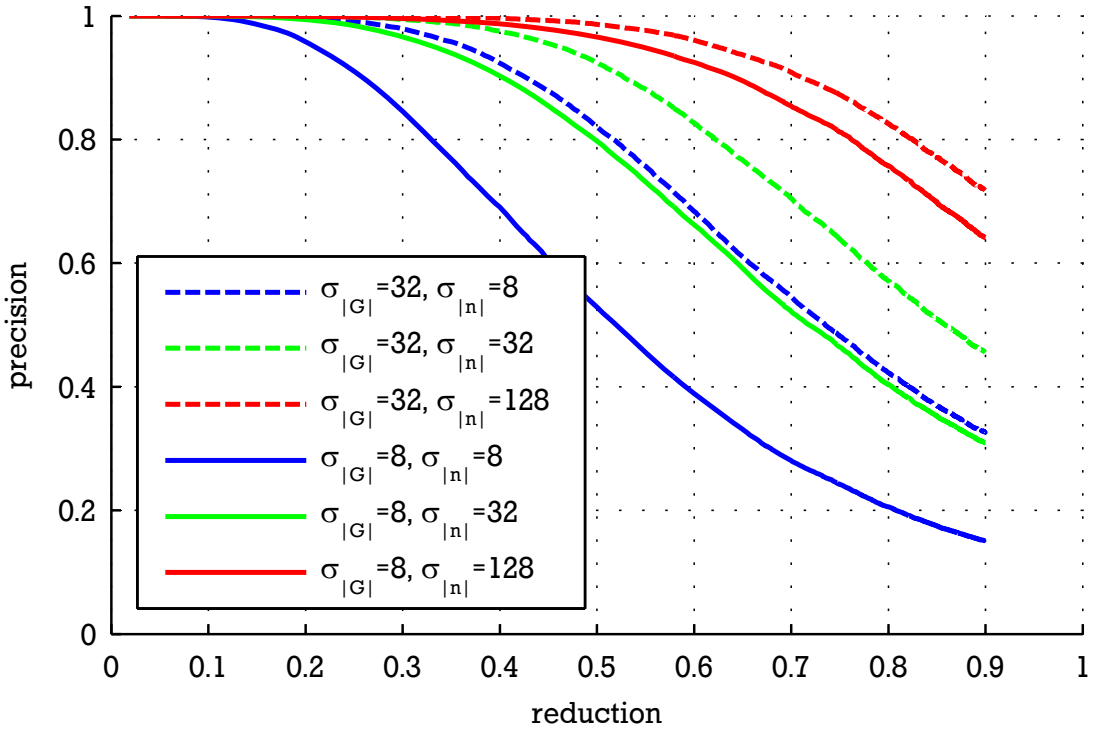


(a) $\overline{d}_1(\mathcal{N}) > \overline{d}_1(\mathcal{G})$

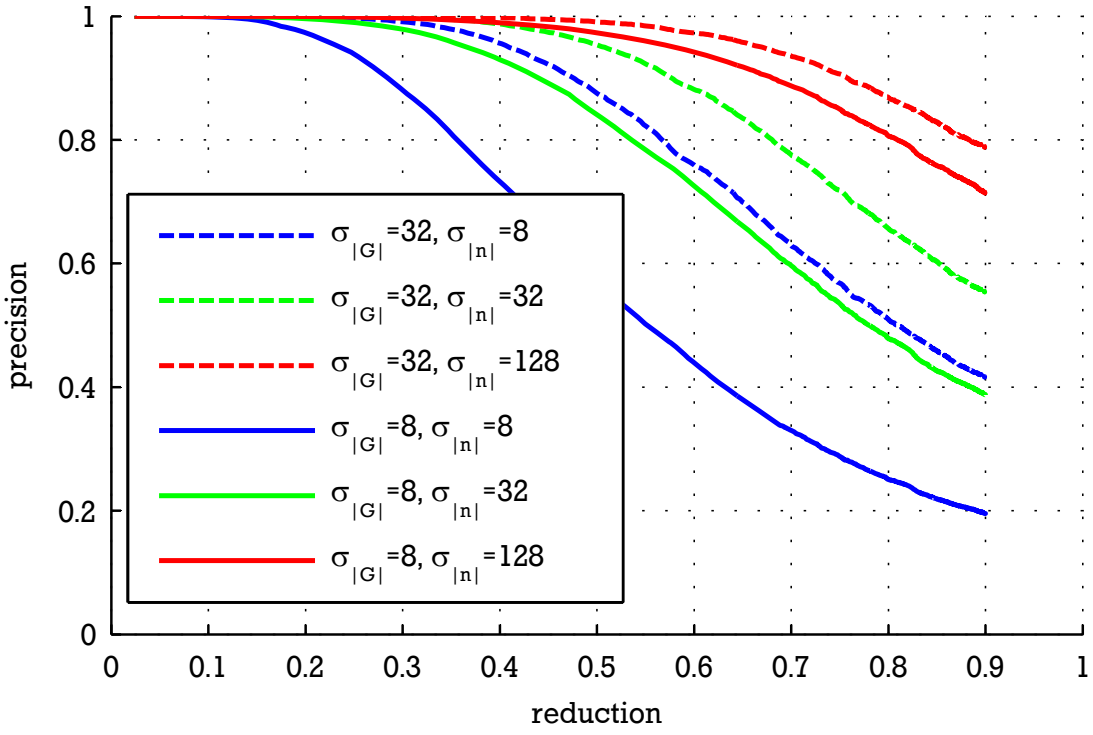


(b) $\bar{s}_2(\mathcal{N}) < \bar{s}_2(\mathcal{G})$

Figure 5.22.: Comparison of the regular distance penalty score d with the normalized distance penalty score d_{norm} using notification sets with dimension characteristics different from the filter set



(a) uniform distribution



(b) clustered distribution

Figure 5.23.: Results for different sets of 1-interval filters $\mathscr{G}$ and notifications $\mathscr{N}$ with varying sizes and distribution pattern

Figure 5.23, reduction values of 15–50% could thus be achieved without significantly reducing precision.

Filter Set Size

In a final experiment, we investigated the impact of the size of the input filter set $\mathscr{G}$ on the measured characteristics. The experimental algorithm was changed slightly: The penalty score threshold was constant[13], but the number of filters in the reduced set $\mathscr{F}$ changed. In an initial run, $|\mathscr{G}| = 500$ filters were added to or merged into $\mathscr{F}$. In subsequent runs with 200 new filters in $\mathscr{G}$ in each run, the filters were continuously added to or merged into $\mathscr{F}$ without re-initializing $\mathscr{F}$. Reduction and precision were measured after each run, thus generating results that can be interpreted as if the original experimental algorithm was executed for input filter sets with different sizes.

Figure 5.24 shows a sample of the results[14] in plots of precision and reduction over input filter set size. In all results of these experiments, we found that reduction increases and precision decreases with increasing input set size.

The reason is that it is obviously more likely for a filter G to be in the proximity of a filter $F \in \mathscr{F}$ (below the distance threshold) if there are more such filters distributed in the bounded filter space. Hence, the probability that a filter G is merged with a filter $F \in \mathscr{F}$ depends considerably on the number of potential merging candidates, i.e., the size of $\mathscr{F}$ at the moment G is processed; the larger $|\mathscr{F}|$ the higher the chance that there is an eligible merging candidate for G, and the more mergers are created.

Since every created merger potentially negatively influences filtering quality, precision decreases with larger numbers of created mergers. The reduction depends directly on the relative number of created mergers and therefore increases with increasing $|\mathscr{G}|$.

5.4.6 Cover Probability

Filter merging is not the only cause for reduction, but also cover relations among filters. Since our experimental algorithm drops a filter G if it is covered by any $F \in \mathscr{F}$ before attempting to merge it, every such occurence of a cover relation adds to the filter set reduction. We therefore also examined this effect. In another run of the experiments, the *cover probability* c was determined for different threshold values and differently large input filter sets.

[13] The penalty score threshold for each experiment was chosen to be 20 times the added normalized per-dimensional mean distance of the sample notification set (with $|\mathscr{N}| = 10,000$), $p_0 = 20 \cdot \left(k_1 \overline{d}_1(\mathscr{N}) + k_2 \overline{d}_2(\mathscr{N})\right)$, thus choosing a fix value that relates to the notification characteristics.

[14] Filters and notifications were generated from the set $\mathscr{R}_4$.

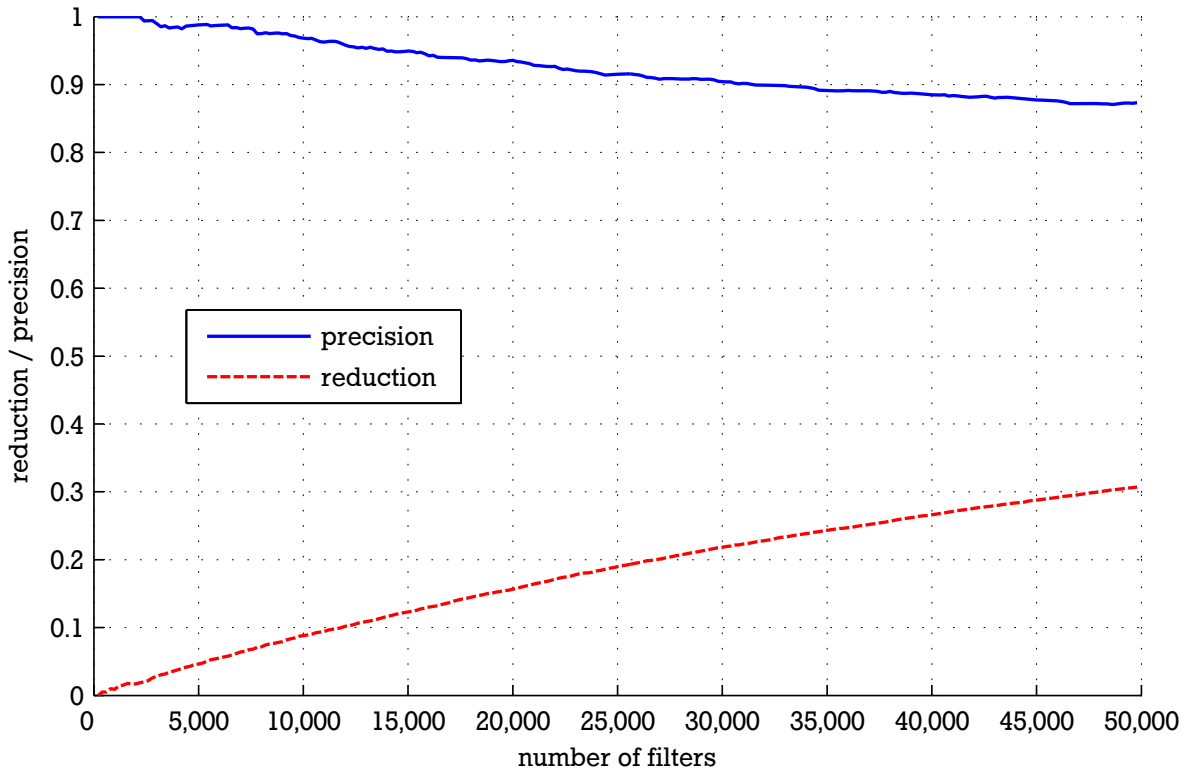


Figure 5.24.: Results (reduction and precision) for differently large input filter sets $\mathscr{G}$

Figure 5.25 shows a sample of the results for varying filter set sizes.[15] The cover probability $c(\mathscr{F})$ was determined by counting in each of the runs described above the number of times a filter G was dropped because it was covered by some $F \in \mathscr{F}$. If that happens for instance 10 times in the 20th run of 200 filters, $c = \frac{10}{200}$ for an input set size between 4,300 and 4,500 filters (with 500 filters in the initial set).

The cover probability $c(F)$ is the probability that F covers a random filter, and $c(\mathscr{F})$ is the probability that some $F \in \mathscr{F}$ covers a random filter. If we denote with $\bar{c}$ the average cover probability $c(F)$ of the filters in $\mathscr{F}$, $c(\mathscr{F})$ is given by $1-(1-\bar{c})^{|\mathscr{F}|}$. Hence, we can expect $c(\mathscr{F})$ to increase less than linearly with the size of $\mathscr{F}$. In addition, the number of filters in $\mathscr{F}$ also increases less than linearly with the size of the input set $\mathscr{G}$ because of increasing reduction. Nevertheless, the cover probability curve exhibits a more than linear slope, which can only be attributed to increasing $\bar{c}$ because of higher cover probability of merger filters. The reason for this is the same as for the decreased filtering quality: Imperfect mergers are larger than the added sizes of its constituents

[15] The plot is smoothed using a moving average lowpass filter. This was necessary because the number of cover occurences is very low, which leads to a heavily oscillating plot because of statistically normal meanderings of the number of occurences so that the un-smoothed plot hardly reveals the important information.

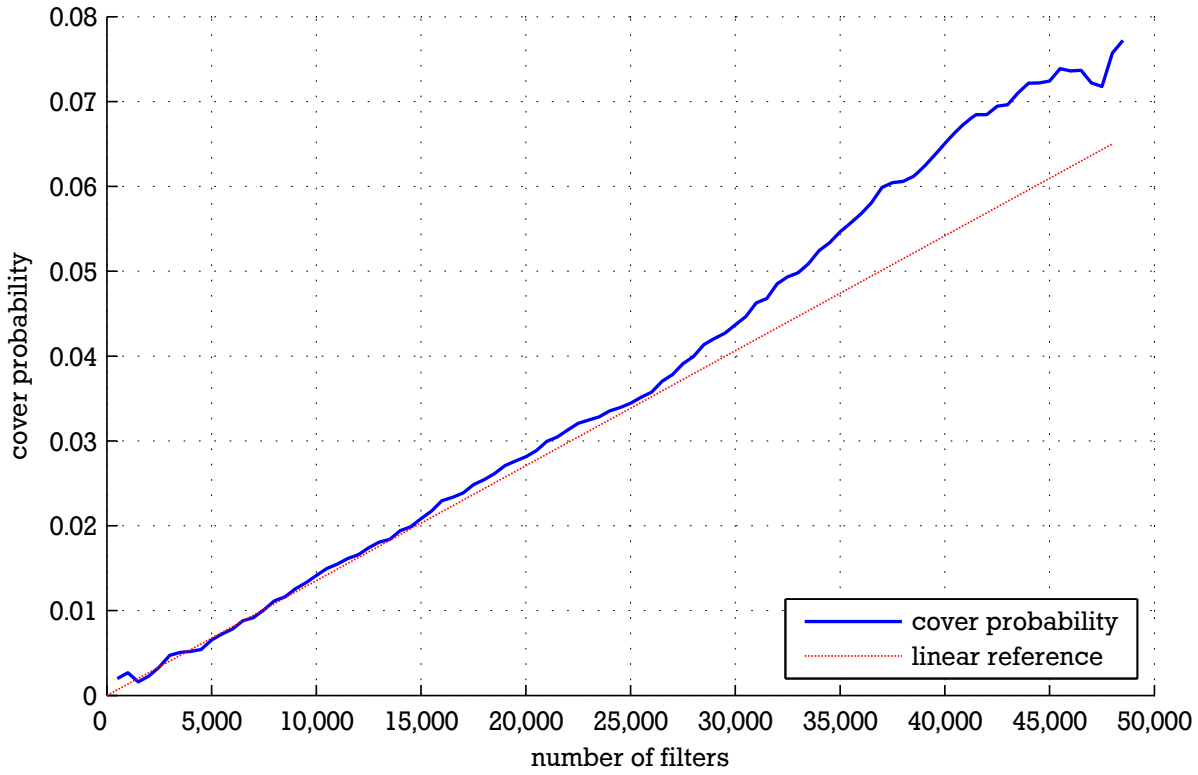


Figure 5.25.: Results (cover probability) for differently large input sets $\mathscr{G}$

and thus cover more of the filter space, which leads to a higher cover probability of a merger filter in relation to its constituents.

5.5 Conclusion

This chapter presented a discussion of filter merging. Filter merging operations were first introduced for the filter types required by our application, spatial and interval filters. Next, the general goal of filter merging was defined generically, and the filter merging problem was stated as a trade-off between filtering quality (measured in *precision*) and filter numbers (measured in *reduction*). A heurstical approach to the problem was presented that builds on estimating the quality of a potential merger of two filters before the merging is actually carried out. The general idea of merger quality estimation is that a potential merger is the better the more similar its constituting filters are. Filter similarity was defined for this purpose as the likelihood that the filters match the same notifications. Two simple measures were defined and used to describe this similarity: filter distance and filter size, based on which different *penalty score functions* were introduced. For two arbitrary filters, these functions return a numerical value for

the estimated merger quality, represented as an inverse *penalty* score such that higher values mean worse estimated mergers.

The approach was evaluated in experiments using a wide range of different sets of filters and notifications. The penalty score functions were compared, and effects of filter and notification characteristics were discussed based on the experimental results. It turned out that the penalty score function based on *normalized filter distance* outperformed all other functions. It is based on the assumption that proximity of filters in the filter space is a useful indication of their similarity, and defines filter distance as the L_1 distance between the filter centroids in a normalized, virtual filter space that results from stretching dimensions individually by a factor calculated from per-dimension mean distance and mean size of notifications, thus requiring a previous analysis of a sample set of notifications.

Higher filtering quality could be achieved at any target reduction with this approach than with the penalty score functions based on regular filter distance and those based on filter size. The latter one additionally exhibited a major drawback because of their definition that evaluates merger size in relation to the sizes of the original filters. This relative approach favors the emergence of merger clusters leading to nosediving precision, and thus degrading the overall approach. The penalty score based on normalized distance however turned out to provide good and stable results in the overall filter merging approach.

Filter and notification characteristics considerably impact the achievable results. If filters and/or notifications are distributed densely in the filter space, filtering quality is generally less reduced by filter merging. This is the case especially with clustered data, as one would expect when interests (represented as filters) and event occurences (represented as notifications) exhibit a strong locality, i.e., if there are many filters for approximately the same “area”, these can be merged without considerably reducing filtering quality.

Varying the size of the input data sets at constant penalty score threshold, we found that the relative reduction is generally higher with a higher number of input filters, which is not surprising because it is obviously more likely to find a merging candidate for a given filter G in a larger set of filters $\mathscr{F}$ than in a smaller one. Since the precision of the reduced set decreases with the number of created mergers, larger input sets generally result in filter sets with lower precision. Finally, we also found that the cover probability, i.e., the likelihood that a given filter G is covered by some filter $F \in \mathscr{F}$ does not only depend on the size of $\mathscr{F}$, but also on the number of merger filters in $\mathscr{F}$. This is because imperfect mergers cover a larger section of the filter space than their constituents. These broader filters therefore generally exhibit a higher probability to cover some other filter.

6 Filter Handling Scheme

This chapter presents and discusses the application of filter merging in the filter processing procedures of a broker as an advanced filter handling scheme for the distributed notification service of a pub/sub system. This scheme aims to increase system scalability compared to the simple subscription flooding approach through, firstly, breaking the linear dependency of broker routing table sizes on the number of system-wide active subscriptions by merging routing entries. Secondly, instead of disseminating the original client subscriptions throughout the broker network, the created mergers are forwarded between brokers, and, subsequently, only filters are forwarded that are not covered by previously forwarded ones, thus limiting the extent of subscription dissemination.

Section 6.1 first presents the basic system model, and the fundamental broker functionality. Section 6.2 presents two broker algorithms: Firstly, for processing an incoming control message (containing subscriptions and/or unsubscriptions) and secondly, for compiling and forwarding an approriate control message to the neighbor brokers.

The intended effect of our filter handling scheme is reduction of routing table sizes and filter forwarding overhead. Routing table size reduction is achieved by merging two or more filters into one. Filter forwarding overhead reduction is achieved by exploiting cover relations among filters, for which the probability increases through the creation of broader merger filters. Our imperfect filtering approach however introduces the possibility of notification forwarding overhead through degraded filtering quality. The extent of each of the effects depends in a particular setup on a number of characteristics, which are discussed theoretically in Section 6.3. We derive formulae to calculate the effects based on filter and notification set characteristics, i.e. reduction, precision, and cover probability. In Section 6.4 a particular real-world scenario with real 4-dimensional trajectory subscriptions and appropriate aeronautical events is assumed, for which we determine in experiments the required characteristics to derive statements on the overall benefit of the approach.

6.1 System Model

We assume a slightly different system model than the one presented in Section 2.2. Instead of designated border brokers, clients connect to the notification service through a *local broker* that resides on the client system.[1] The broker network itself consists exclusively of *inner brokers*. The local broker implements the pub/sub interface with

[1] This is also the system model of REBECA, see Appendix A.

the *pub*, *sub*, and *unsub* methods, and the client notification call-back method *notify*. The local broker is connected to exactly one inner broker (Figure 6.1).

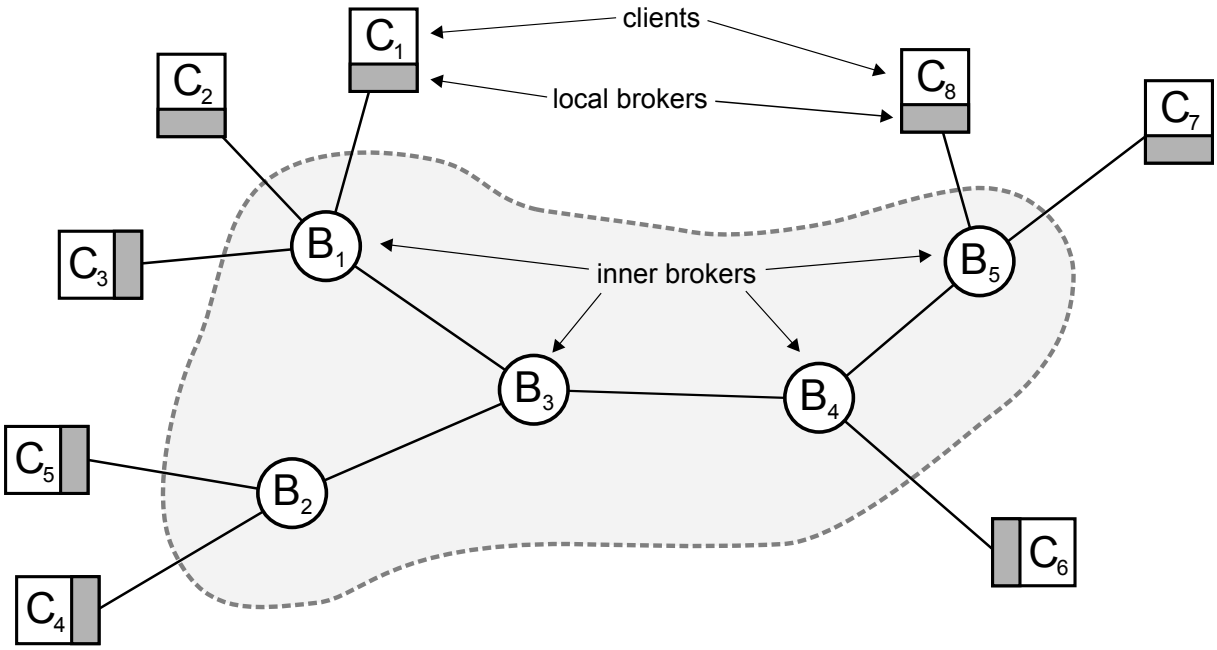


Figure 6.1.: Publish/subscribe system model with clients, local brokers, and inner brokers

Every (local and inner) broker maintains a routing table that consists of routing entries (F, D) that are ordered pairs of a filter F and a destination D. In the following, we denote the routing table with $\mathscr{R} = \{(F, D\}$. A broker forwards a received notification n to a destination D_i if there is an entry (F_i, D_i) in its routing table where F_i matches n, unless D_i is the source S of the notification, i.e., the neighbor broker that forwarded n. The set of all neighbors of a broker is in the following denoted with $\mathscr{D}$. In the case of an inner broker, this set consists of all its neighbor brokers, local and inner. In the case of a local broker, this set consists only of the inner broker the local broker is connected to and the client C. If the routing table contains an entry for the client that matches an incoming notification n, the client is notified by calling *notify*(n). The *process* function in Figure 6.2 takes as input arguments the notification n and the source S the notification was received from.

The advanced filter handling algorithms that are presented in the next sections are applied only when receiving a control message from another broker, whereas the local broker's routing table updates induced by the client's calls of *sub* and *unsub* do not employ such algorithms. Routing entries for the local client C are simply added to or

```
function process(n,S)
begin
    foreach D ∈ {E|(F,E) ∈ ℛ.F(n) = true ∧ E ≠ S} do
        if D = C then
            C ⇒notify(n)
        else
            forward(D,n)
        end
    end
end
```

Figure 6.2.: Pseudocode: Processing of an incoming notification

removed from the routing table. The reason for this is that filter merging and filter simplification lead to imperfect notification filtering. Whereas falsely matched and unnecessarily forwarded notifications can be accepted within the broker network, a client must be notified of all published events that it has previously subscribed for, and only of those. Hence, the delivery of wrong notifications to clients must be avoided.

```
function sub(F)
begin
    ℛ ← ℛ ∪ {(F,C)}
    forward(control({F},∅))
end

function unsub(F)
begin
    ℛ ← ℛ \ {(F,C)}
    forward(control(∅,{F}))
end

function pub(n)
begin
    forward(n)
end
```

Figure 6.3.: Pseudocode: Publish/subscribe interface methods

Figure 6.3 shows the implementation of the pub/sub interface methods. The local brokers provide perfect notification filtering toward the subscriber clients. When a client subscribes to a filter F by calling $sub(F)$, a new routing entry (F,C) is added to the

local broker's routing table (where C denotes the local client), and the broker forwards this subscription in a message $control(\{F\}, \emptyset)$ to the inner broker. Such broker control messages generally take the form $control(\mathscr{S}, \mathscr{U})$, where $\mathscr{S}$ and $\mathscr{U}$ are (possibly empty) sets of subscription filters and unsubscription filters, respectively, for reasons described in the next section. Calls to $unsub(F)$ are handled likewise, and notifications published by calling $pub(n)$ are simply forwarded to the inner broker.

The processing of incoming control messages is the same for all brokers. The algorithms for routing table updates and filter forwarding are described in the next subsection.

6.2 Control Message Handling Algorithms

A broker's processing of a control message received from a neighbor broker consists of two parts: Firstly, the routing table is updated according to the received subscriptions and unsubscriptions. In the approach presented here, the filter merging functionality is encapsulated entirely in the routing table update algorithm described in Section 6.2.1. Secondly, the filters are forwarded to eligible neighbor brokers. When filter mergers are created, there is a chance that a newly received subscription filter is covered by such a merger, in which case it does not have to be forwarded. An algorithm exploiting covering relations among filters is presented in Section 6.2.2.

6.2.1 Routing Table Update

Applying filter merging to reduce filter forwarding overhead requires that the filters that are not forwarded at one point be nevertheless stored by the broker, and potentially be forwarded to its neighbors at a later point, when a merger is disintegrated or a covering subscription filter is canceled by an unsubscription. Therefore, an unsubscription may have to be forwarded with a set of formerly covered (*uncovered*) filters or remaining constituents of a merger. The subscription and unsubscription filters are forwarded in the sets $\mathscr{S}$ and $\mathscr{U}$, respectively, in the broker control message.

The routing table update algorithm, presented in Figure 6.5, takes as input the set of subscription filters $\mathscr{S}$, the set of unsubscription filters $\mathscr{U}$, and the source S of the control message, i.e., a pointer to the communication link with the respective neighbor broker. In the course of processing the filter sets, filters are removed from or added to the sets, which are subsequently forwarded as discussed in the next Section 6.2.2.

The algorithm presented here is based on one originally presented by G. Mühl in his doctoral dissertation [134] for "merging-based routing". Whereas his algorithm was designed for perfect filter mergers only, our algorithm is capable of handling imperfect

mergers equally well by not making a distinction between different kinds of mergers, and assigning the merging decision to the *findMergingCandidate* function introduced before. Furthermore, it adds an enhanced handling of uncovered filters and of the constituents of disintegrated mergers. Our algorithm processes uncovered filters and the constituents of a disintegrated merger like newly received subscriptions, thus first trying to re-merge them, while the original algorithm simply forwarded those filters to neighbor brokers.

In the following, $m(M)$ denotes the set of consituting filters of a merger filter M. Hence, checking for $m(F) = \emptyset$ for any filter F throughout the algorithm is the test whether F is a merger.

Input: Sets of subscripton filters $\mathscr{S}$, unsubscription filters $\mathscr{U}$, their source S
Output: Preprocessed sets of subscripton filters $\mathscr{S}$ and unsubscription filters $\mathscr{U}$

1 **foreach** $F \in \mathscr{U}$ **do**
2 $\quad \mathscr{R} \leftarrow \mathscr{R} \setminus \{(G,D) | D = S \wedge G \sqsubseteq F\}$
3 $\quad$ **foreach** $G \in \{H | (H,S) \in \mathscr{R}\}$ **do**
4 $\quad\quad$ **if** $\exists G_M \in m(G).G_M \sqsubseteq F$ **then**
5 $\quad\quad\quad \mathscr{R} \leftarrow \mathscr{R} \setminus \{(G,S)\}$
6 $\quad\quad\quad \mathscr{U} \leftarrow \mathscr{U} \cup \{G\}$
7 $\quad\quad\quad \mathscr{S} \leftarrow \mathscr{S} \cup \{G_M \in m(G) | G_M \not\sqsubseteq F\}$
8 $\quad\quad$ **end**
9 $\quad$ **end**
10 **end**
11 $\mathscr{S} \leftarrow \mathscr{S} \setminus \{F | \exists G \in \mathscr{S}.G \neq F \wedge G \sqsupseteq F\}$

Figure 6.4.: Pseudocode: Routing table update – processing unsubscriptions

Processing of Unsubscriptions

The set of unsubscriptions is processed first (Figure 6.4). Every routing entry in the routing table $\mathscr{R}$ for the destination S (the source of the control message) that is covered by an unsubscription filter $F \in \mathscr{U}$ is removed from the table (line 2). Next, if an unsubscription filter covers a constituent of a merger in the routing table, this merger is disintegrated (lines 3 through 9). The respective routing entry is removed from the table and the merger is added to the set of unsubscriptions $\mathscr{U}$. Its remaining constituents (those, that are not covered by an unsubscription filter) are added to the set of subscription filters $\mathscr{S}$. In this step, filters may be added to the sets that cover or are covered

Input: Preprocessed sets of subscripton filters $\mathscr{S}$, unsubscription filters $\mathscr{U}$, source S
Output: Sets of subscripton filters $\mathscr{S}$ and unsubscription filters $\mathscr{U}$ to forward

1 **foreach** $F \in \mathscr{S}$ **do**
2 $\quad \mathscr{R} \leftarrow \mathscr{R} \setminus \{(G,D) | D = S \wedge G \sqsubseteq F\}$
3 $\quad$ **foreach** $G \in \{H | (H,S) \in \mathscr{R}\}$ **do**
4 $\quad\quad$ **if** $G \sqsupseteq F$ **then**
5 $\quad\quad\quad m(G) \leftarrow m(G) \cup F$
6 $\quad\quad\quad \mathscr{S} \leftarrow \mathscr{S} \setminus \{F\}$
7 $\quad\quad\quad$ **continue** F
8 $\quad\quad$ **end**
9 $\quad\quad$ **if** $\exists G_M \in m(G).G_M \sqsubseteq F$ **then**
10 $\quad\quad\quad \mathscr{R} \leftarrow \mathscr{R} \setminus \{(G,S)\}$
11 $\quad\quad\quad \mathscr{U} \leftarrow \mathscr{U} \cup \{G\}$
12 $\quad\quad\quad \mathscr{S} \leftarrow \mathscr{S} \cup \{G_M \in m(G) | \nexists C \in \mathscr{S}.C \sqsupseteq G_M\}$
13 $\quad\quad$ **end**
14 $\quad$ **end**
15 $\quad F_{\text{mc}} \leftarrow$`findMergingCandidate`$(\{G | (G,S) \in \mathscr{R}\}, F)$
16 $\quad$ **if** $F_{\text{mc}} = null$ **then**
17 $\quad\quad \mathscr{R} \leftarrow \mathscr{R} \cup \{(F,S)\}$
18 $\quad$ **else**
19 $\quad\quad M = F \sqcup F_{\text{mc}}$
20 $\quad\quad$ **if** $m(F_{\text{mc}}) = \emptyset$ **then**
21 $\quad\quad\quad m(M) \leftarrow \{F, F_{\text{mc}}\}$
22 $\quad\quad$ **else**
23 $\quad\quad\quad m(M) \leftarrow \{F\} \cup m(F_{\text{mc}})$
24 $\quad\quad$ **end**
25 $\quad\quad$ **foreach** $G \in \{H | (H,S) \in \mathscr{R} \wedge M \sqsupseteq H\}$ **do**
26 $\quad\quad\quad$ **if** $m(G) = \emptyset$ **then** $m(M) \leftarrow m(M) \cup G$ **else** $m(M) \leftarrow m(M) \cup m(G)$
27 $\quad\quad$ **end**
28 $\quad\quad \mathscr{R} \leftarrow \mathscr{R} \cup \{(M,S)\}$
29 $\quad\quad \mathscr{S} \leftarrow \mathscr{S} \cup \{M\} \setminus F$
30 $\quad$ **end**
31 **end**
32 $\mathscr{U} \leftarrow \mathscr{U} \setminus \{F | \exists G \in \mathscr{U}.G \neq F \wedge G \sqsupseteq F\}$

Figure 6.5.: Pseudocode: Routing table update – processing subscriptions

by a different filter in the set. The covered filters are therefore removed from the set of subscriptions (line 11) before it is processed.

Processing of Subscriptions

The preprocessed set of subscription filters is processed next (Figure 6.5). For every subscription filter F, all routing entries that are covered by this filter are first removed from the routing table (line 2). Next, if there is an appropriate covering merger in the routing table, F is simply added to the set of its constituents and removed from $\mathscr{S}$ (lines 4 through 8). Note that a covering filter for F can only be a previously created merger, because F would not have been forwarded by S if a covering filter had been forwarded previously. If this succeeds, the next filter $F \in \mathscr{S}$ is processed. Otherwise, it is checked if F covers a constituent of a merger in the routing table, in which case the merger is disintegrated (lines 9 through 13). The difference to the merger disintegration because of an unsubscription before is that the remaining constituents are added to the set of subscriptions only if there is no covering filter in this set (line 12). This may be the case if that covering filter is a merger filter created at the source S or before.

The actual merging functionality sets in next by trying to find a merging candidate for F (line 15). If no such filter F_{mc} is found, F is simply added to the routing table in a new routing entry (F,S). Otherwise, F is merged with F_{mc} to M and F as well as F_{mc} or, in the case that F_{mc} already is a merger, the constituents of it, are added to the set of constituents of M (lines 19 through 24). All routing entries that are covered by the newly created merger M are then removed from the routing table and the filters or their constituents are added to the set of constituents of M (line 26). Finally, a routing entry for M is added to the routing table and M is added to the set of subscription filters instead of F.

All covered filters that may have been added in the process of disintegrating a merger are removed from the set of unsubscriptions (line 32) before the two filter sets are further processed in the filter forwarding algorithm described next.

6.2.2 Filter Forwarding

The basic idea for exploiting cover relations among filters to reduce filter forwarding overhead is as follows. If there is a routing entry (F_1, D_1) in the routing table, F_1 has been forwarded previously to all neighbors except D_1. If then a subscription filter F_2 is processed (e.g., from neighbor D_2) that is covered by F_1, it is stored in the routing table as (F_2, D_2) but has to be forwarded only to D_1 because all other neighbors have already received F_1 and will therefore forward all notifications matching D_2 anyway. All subsequently received filters F_i that are covered by F_1 and F_2 do not have to be forwarded at

all. What makes this approach more complicated is the handling of such covered filters in the case that the covering filters are canceled, i.e., when in this example an unsubscription for F_1 is received, because then some of the filters F_2, F_3, F_i are uncovered and will have to be forwarded to appropriate neighbors.

The filter forwarding algorithm (Figure 6.6) is in the following explained in detail. It is based on one originally developed by G. Mühl for "covering-based routing". We adapted it for our purpose by making it compatible with the preprocessed filter sets returned by the preceding routing table update algorithm.

The algorithm takes as input the preprocessed sets of subscription and unsubscription filters $\mathscr{S}$ and $\mathscr{U}$, and the source S of the control message. It returns two sets $\mathscr{M}_S, \mathscr{M}_U$ of *filter forwarding records* (F, D) consisting of a filter F and a destination D. The filters are subscription filters in $\mathscr{M}_S$ and unsubscription filters in $\mathscr{M}_U$ and the destinations are pointers to the communication links with neighbor brokers. These records are assumed to be subsequently processed such that a control message $control(\mathscr{S}_D, \mathscr{U}_D)$ is sent to every neighbor D with $\mathscr{S}_D = \{F|(F,D) \in \mathscr{M}_S\}$ and $\mathscr{U}_D = \{F|(F,D) \in \mathscr{M}_U\}$ unless both sets $\mathscr{S}_D$ and $\mathscr{U}_D$ are empty.

Input: Set of subscripton filters $\mathscr{S}$, set of unsubscription filters $\mathscr{U}$, source S
Output: Sets of filter forwarding records $\mathscr{M}_S, \mathscr{M}_U$

if $\mathscr{U} = \emptyset$ **then**
  return $\big($`forwardSubsOnly`$(\mathscr{S}, S), \emptyset\big)$
else
  return `forwardSubsAndUnsubs`$(\mathscr{S}, \mathscr{U}, S)$
end

Figure 6.6.: Pseudocode: Filter forwarding

Forwarding of Subscriptions

If there are no unsubscription filters to process, the eligible destinations for each subscription filter are identified by the logic explained above (Figure 6.7). The routing entry set $\mathscr{R}_1$ consists of all routing table entries that cover F (line 3). Since F has been added to the routing table already (or merged with another filter), there is in any case a covering routing entry (G, S) with either $F = G$ or $F \in m(G)$. If there is only this one covering entry, F is forwarded to all destinations except S (line 5). If there is another covering routing entry, F is forwarded to the destination in this entry (line 7). If there are more than two covering entries, F is not forwarded to any neighbor.

Input: Set of subscripton filters $\mathscr{S}$, source S
Output: Set of subscription filter forwarding records $\mathscr{M}_S$

1 $\mathscr{M}_S \leftarrow \emptyset$
2 **foreach** $F \in \mathscr{S}$ **do**
3 $\quad \mathscr{R}_1 \leftarrow \{(G,D) \in \mathscr{R} | G \sqsupseteq F\}$
4 $\quad$ **if** $\left|\mathscr{R}_1\right| = 1$ **then**
5 $\quad\quad \mathscr{M}_S \leftarrow \mathscr{M}_S \cup \{(F,D) | D \in \mathscr{D} \setminus S\}$
6 $\quad$ **else if** $\left|\mathscr{R}_1\right| = 2$ **then**
7 $\quad\quad \mathscr{M}_S \leftarrow \mathscr{M}_S \cup \{(F,D) | (G,D) \in \mathscr{R}_1 \wedge G \neq F \wedge \nexists G_M \in m(G).F = G_M\}$
8 $\quad$ **end**
9 **end**
10 **return** $\mathscr{M}_S$

Figure 6.7.: Pseudocode: Filter forwarding – function `forwardSubsOnly`

Forwarding of Unsubscriptions, Subscriptions and Formerly Covered Filters

If the set of unsubscriptions is not empty, the same logic is applied to determine the eligible forwarding destinations for each unsubscription filter (Figure 6.8). The set of subscription filters must however be processed differently, because there may be uncovered subscription filters that have to be processed as well. Possibly uncovered subscriptions are all filters in the routing table for which there is a properly covering unsubscription filter in $\mathscr{U}$. All respective routing entries are stored in a temporary set $\mathscr{R}_2$ and all subscription filters $F \in \mathscr{S}$ are added to this set as routing entries (F,S) (line 11). For each of the filters of the routing entries in this set, a similar logic as before is applied for the determination of eligible filter forwarding destinations in a temporary set $\mathscr{D}_1$, which takes only properly covering routing entries into account (lines 13 through 20).

However, the destination D of the currently processed routing entry remains in the set $\mathscr{D}_1$ only if there is another entry in $\mathscr{R}_2$ with an equal filter (lines 21 through 23). This is the case if that other routing entry is a formerly covered filter from S. Then F must be forwarded to D. Otherwise, F came from D and must not be forwarded.

Finally, a filter forwarding record for each $D_1 \in \mathscr{D}_1$ is added to $\mathscr{M}_S$ except for the source S of the original control message (line 24).

Input: Set of subscripton filters $\mathscr{S}$, set of unsubscription filters $\mathscr{U}$, source S
Output: Sets of filter forwarding records $\mathscr{M}_S, \mathscr{M}_U$

1 $\mathscr{M}_S \leftarrow \emptyset$
2 $\mathscr{M}_U \leftarrow \emptyset$
3 **foreach** $F \in \mathscr{U}$ **do**
4 $\quad \mathscr{R}_1 \leftarrow \{(G, D) \in \mathscr{R} | G \sqsupseteq F\}$
5 $\quad$ **if** $\left|\mathscr{R}_1\right| = 1$ **then**
6 $\quad\quad \mathscr{M}_U \leftarrow \mathscr{M}_U \cup \{(F, D) | D \in \mathscr{D} \setminus S\}$
7 $\quad$ **else if** $\left|\mathscr{R}_1\right| = 2$ **then**
8 $\quad\quad \mathscr{M}_U \leftarrow \mathscr{M}_U \cup \{(F, D) | (G, D) \in \mathscr{R}_1 \wedge G \neq F \wedge \nexists G_M \in m(G).F = G_M\}$
9 $\quad$ **end**
10 **end**
11 $\mathscr{R}_2 \leftarrow \{(F, D) | \exists G \in \mathscr{U}.G \sqsupset F\} \cup \{(F, S) | F \in \mathscr{S}\}$
12 **foreach** $(F, D) \in \mathscr{R}_2$ **do**
13 $\quad \mathscr{R}_1 \leftarrow \{(G, D_1) \in \mathscr{R} | G \sqsupset F\}$
14 $\quad$ **if** $\left|\mathscr{R}_1\right| = 0$ **then**
15 $\quad\quad \mathscr{D}_1 \leftarrow \mathscr{D}$
16 $\quad$ **else if** $\left|\mathscr{R}_1\right| = 1$ **then**
17 $\quad\quad \mathscr{D}_1 \leftarrow \mathscr{D} \setminus \{D_1 | (G, D_1) \in \mathscr{R}_2\}$
18 $\quad$ **else**
19 $\quad\quad \mathscr{D}_1 \leftarrow \emptyset$
20 $\quad$ **end**
21 $\quad$ **if** $\nexists G.(G, D_1) \in \mathscr{R}_2 \wedge G \neq F \wedge G \equiv F$ **then**
22 $\quad\quad \mathscr{D}_1 \leftarrow \mathscr{D}_1 \setminus D$
23 $\quad$ **end**
24 $\quad \mathscr{M}_S \leftarrow \mathscr{M}_S \cup \{(F, D_1) | D_1 \in \mathscr{D}_1 \setminus S\}$
25 **end**
26 **return** $\mathscr{M}_S, \mathscr{M}_U$

Figure 6.8.: Pseudocode: Filter forwarding – function `forwardSubsAndUnsubs`

6.3 Theoretical Evaluation

The intrinsic characteristics of a filter handling scheme, by which it can be compared with alternative schemes, are routing table size reduction and resulting forwarding overheads. These effects depend in a particular pub/sub system on a number of factors: the characeristics of the filters and notifications handled by the brokers, the broker network topology and the distribution of producer and consumer clients in the network as well as their activity, i.e., the frequency of subscriptions, unsubscriptions, and publications. In this section, the evaluation of these effects is approached theoretically by deriving formulae based on intrinsic characteristics of the filter and notification sets, namely reduction, precision and cover probability.

Subsection 6.3.1 first discusses the distribution of active subscription filters in the broker network in a static view of a distributed notification service. In the subsequent subsections, the effects of our filter handling scheme are then formally discussed. For this discussion, Subsection 6.3.2 first formally introduces derived characteristics of filters and filter sets, namely *matching probability* and the previously mentioned *cover probability*. In Subsection 6.3.3, routing table size reduction and forwarding overheads are formally described with respect to an isolated broker-to-broker link, and in Subsection 6.3.4 the propagation of the effects throughout the broker network is discussed.

6.3.1 Subscription Filter Distribution

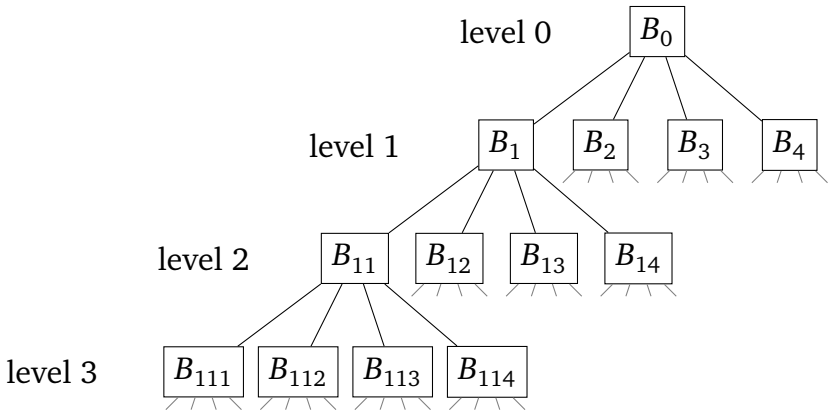


Figure 6.9.: Example broker network topology

If a regular, cycle-free topology of the broker network is assumed, brokers are organized in a hierarchy with the border brokers at the leaf level and a *root broker* at the root level of the tree. Figure 6.9 shows a section of an example topology with four subordinate brokers at each level. The root level is denoted with *level 0*. The figure shows

only the hierarchy path $B_0 \rightarrow B_1 \rightarrow B_{11}$ and all their neighbor brokers, the other subnetworks are not shown in the figure. We assume for simplicity a uniform distribution of producer and consumer clients to border brokers, and a uniform distribution of active subscriptions among the clients.

Let $\mathscr{R}_1^{\rightarrow 0}$ denote the set of all subscription filters of all routing entries for B_0 in B_1's routing table, $\mathscr{R}_1^{\rightarrow 0} = \{F | (F, B_0) \in \mathscr{R}_1\}$. Each broker holds in its routing tables (potentially merged) subscription filters for each neighbor broker, i.e., for its superordinate and its four subordinate brokers, e.g., broker B_{11} holds a routing table $\mathscr{R}_{11}$ consisting of subsets of routing entries $\mathscr{R}_{11}^{\rightarrow 1}$ for the root broker B_1, $\mathscr{R}_{11}^{\rightarrow 111}$ for the subordinate broker B_{111}, $\mathscr{R}_{11}^{\rightarrow 112}$ for B_{112}, $\mathscr{R}_{11}^{\rightarrow 113}$ for B_{113}, and $\mathscr{R}_{11}^{\rightarrow 114}$ for B_{114}, $\mathscr{R}_{11} = \left\{\mathscr{R}_{11}^{\rightarrow 1}, \mathscr{R}_{11}^{\rightarrow 111}, \mathscr{R}_{11}^{\rightarrow 112}, \mathscr{R}_{11}^{\rightarrow 113}, \mathscr{R}_{11}^{\rightarrow 114}\right\}$. Filter merging can only take place among members of these routing table subsets, because only routing entries for the same destination can be merged.

Note that every active subscription from any consumer client of the entire pub/sub system is represented (possibly merged with another subscription) in every broker routing table, in exactly one of these subsets. A subset thus contains (a representation of) the active subscriptions in the subnetwork "behind" the respective neighbor broker. Since these subnetworks are differently large, the routing table subsets are differently "large", i.e., with more or less subscriptions represented in them.[2] The largest ones of these routing table subsets are found in the border brokers towards their superordinate brokers. In the topology section depicted in the figure, the largest routing table subsets would be $\mathscr{R}_{111}^{\rightarrow 11}$, $\mathscr{R}_{112}^{\rightarrow 11}$, $\mathscr{R}_{113}^{\rightarrow 11}$, and $\mathscr{R}_{114}^{\rightarrow 11}$. The distribution of all system-wide active subscriptions in the subsets of $\mathscr{R}_0$ is even, i.e., the same number of subscriptions is represented in $\mathscr{R}_0^{\rightarrow 1}$, $\mathscr{R}_0^{\rightarrow 2}$, $\mathscr{R}_0^{\rightarrow 3}$, and $\mathscr{R}_0^{\rightarrow 4}$.

In the following subsections, we discuss the reduction of these sets and the resulting forwarding overhead effects with respect to saved filter forwarding and additional notification forwarding, before returning to an evaluation of the system-wide effects in Subsection 6.3.4, where the propagation of forwarding overheads is discussed.

6.3.2 Derived Filter Characteristics

Two more characteristics of filters and filter sets shall be formally introduced and defined in the following, which are required in the discussion of filter handling scheme characteristics: the matching probability m and the cover probability c.

[2] Applying the merging-based filter handling scheme, "larger" subsets in this sense do not necessarily have to contain more elements because of potentially different reduction.

Matching Probability

Let $\mathbb{N}$ denote the set of all notifications, and $N(F)$, as before, the set of all notifications matched by F. We define the *matching probability* of a filter as the ratio between the number of notification it matches and the number of all possible notifications:

$$m(F) = \frac{|N(F)|}{|\mathbb{N}|}$$

It is the probability by which F matches a random notification $n \in \mathbb{N}$. Since this value will in practice often not be computable because the numbers $|N(F)|$ and $|\mathbb{N}|$ are unknown, the matching probability can be estimated using a large enough sample set of notifications $\mathcal{N}$:

$$\hat{m}_{\mathcal{N}}(F) = \frac{|\mathcal{N} \cap N(F)|}{|\mathcal{N}|}.$$

Then $\hat{m}_{\mathcal{N}}(F)$ is the probability by which F matches a random notification $n \in \mathcal{N}$.

Accordingly, we define the matching probability of a filter set $\mathscr{F}$ as the fraction of notifications matched by any $F \in \mathscr{F}$. It is given by

$$m(\mathscr{F}) = \frac{|N(\mathscr{F})|}{|\mathbb{N}|}.$$

It is the probability by which some $F \in \mathscr{F}$ matches a random notification $n \in \mathbb{N}$. Again, the matching probability can be estimated using a sample set of notifications $\mathcal{N}$:

$$\hat{m}_{\mathcal{N}}(\mathscr{F}) = \frac{|\mathcal{N} \cap N(\mathscr{F})|}{|\mathcal{N}|}. \tag{6.1}$$

where $\mathcal{N} \cap N(\mathscr{F})$ denotes the set of all notifications $n \in \mathcal{N}$ that are matched by any of the filters in $\mathscr{F}$, $\mathcal{N} \cap N(\mathscr{F}) = \{n \in \mathcal{N} | \exists F \in \mathscr{F}.F(n)\}$.

Cover Probability

Lef $\mathbb{F}$ denote the set of all filters, and let $F \in \mathbb{F}$. The cover probability c of F is defined as the fraction of filters $G \in \mathbb{F}$ covered by F:

$$c(F) = \frac{|\{G \in \mathbb{F} | F \sqsupseteq G\}|}{|\mathbb{F}|}.$$

As before, this value can be estimated using a sample filter set $\mathscr{G}$. It is then given by

$$\hat{c}_{\mathscr{G}}(F) = \frac{|\{G \in \mathscr{G} | F \sqsupseteq G\}|}{|\mathscr{G}|}.$$

Note that in Section 5.4.6, we had already determined the cover probability experimentally in much the same approach.[3]

According to the filter cover probability, we define the cover probability of a filter set $\mathscr{F}$ as the probability that any $F \in \mathscr{F}$ covers a random $G \in \mathbb{F}$:

$$c(\mathscr{F}) = \frac{|\{G \in \mathbb{F} | \exists F \in \mathscr{F}.F \sqsupseteq G\}|}{|\mathbb{F}|}.$$

The estimation of $c(\mathscr{F})$ with a sample set of filters $\mathscr{G}$ is given by

$$\hat{c}_{\mathscr{G}}(\mathscr{F}) = \frac{|\{G \in \mathscr{G} | \exists F \in \mathscr{F}.F \sqsupseteq G\}|}{|\mathscr{G}|}. \tag{6.2}$$

6.3.3 Filter Handling Scheme Characteristics

In order to describe the effects of our filter handling scheme (routing table size reduction, filter forwarding overhead, and notification forwarding overhead) in numerical values actually expressing the achieved benefit, we assume the subscription flooding approach as baseline, i.e., routing table size and forwarding overheads are measured and expressed relatively to the respective values in this simple approach.

We consider in the following a single broker-to-broker connection, i.e., two nodes and the connecting edge in the network graph, as depicted in Figure 6.10.

[3] There, $F \in \mathscr{G}$ as well. This changes the definition slightly, because it would be of no sense including F itself in the sample filter set used to evaluate its cover probability. It is therefore changed to $\hat{c}_{\mathscr{G}}(F) = \frac{|\{G \in \mathscr{G} | F \sqsupseteq G\}|}{|\mathscr{G} \setminus F|}$.

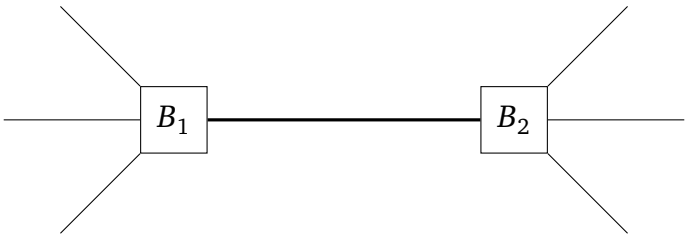


Figure 6.10.: Broker-broker connection: Two nodes and one edge of the broker network graph

Routing Table Size Reduction

Let $\mathscr{F}$ be the set of all filters of active subscriptions of consumer clients in the sub-network "behind" B_2 as seen from B_1. In the subscription flooding approach, $\mathscr{R}_1^{\rightarrow 2} = \mathscr{F}$. Applying our filter handling scheme, the filters in $\mathscr{F}$ are merged as they are propagated along the broker tree in the sub-network, and $\mathscr{R}_1^{\rightarrow 2}$ finally contains only a reduced number of all filters. Equation (5.2) gives the reduction $r_{\mathscr{F}}\left(\mathscr{R}_1^{\rightarrow 2}\right)$ of the routing table subset.

Notification Forwarding Overhead

We define notification forwarding overhead as the number of additionally (unnecessarily) forwarded notifications resulting from the imperfect filtering introduced by imperfectly merging filters.

The notification forwarding overhead is obviously easily expressed through the precision p of the reduced filter set $\mathscr{R}_1^{\rightarrow 2}$. $p\left(\mathscr{R}_1^{\rightarrow 2}\right)$ can be calculated according to Equation (5.3). The set of sample notifications required in the equation is the set of all notifications that are published somewhere in the sub-network "behind" B_1 as seen from B_2. If we denote this notification set with $\mathscr{N}_1^{2\leftarrow}$, the precision of the reduced filter set is given by $p_{\mathscr{N}_1^{2\leftarrow}}\left(\mathscr{R}_1^{\rightarrow 2}\right)$.

To determine the number of unnecessarily forwarded notifications, the matching probability m of the filter set $\mathscr{F}$ is required. It can be "estimated" as $m_{\mathscr{N}_1^{2\leftarrow}}(\mathscr{F})$. If a number of N notifications are published, perfect filtering would lead to the forwarding of $N \cdot m$ notifications (true positives). In our filter handling scheme $N \cdot m/p$ notifications are forwarded (all positives). The forwarding overhead introduced (false positives) is thus $N \cdot m/p - N \cdot m$, or generally, for every notification $n \in \mathscr{N}_1^{2\leftarrow}$, the

notification forwarding overhead (increase) O_n introduced by the imperfect filtering of $\mathscr{R}_1^{\rightarrow 2}$ is

$$O_n\left(B_1 \rightarrow B_2\right) = \frac{m(\mathscr{F})}{p\left(\mathscr{R}_1^{\rightarrow 2}\right)} - m(\mathscr{F}). \tag{6.3}$$

This many notifications are unnecessarily forwarded from B_1 to B_2 as the result of the publication of one notification in the sub-network behind B_1.

Filter Forwarding Overhead

We define filter forwarding overhead as the average number of control messages sent from B_1 to B_2 as the result of a consumer's subscription somewhere in the sub-network behind B_1 as seen from B_2. For subscription flooding this number is constant 1.

In our filter handling scheme, a subscription filter is only forwarded unless a covering filter has previously been forwarded. All previously forwarded filters are represented in $\mathscr{R}_2^{\rightarrow 1}$.[4] Hence, the probability by which a subscription filter is forwarded depends on the cover probability c of this routing table subset. The number of filters forwarded is $1 - c\left(\mathscr{R}_2^{\rightarrow 1}\right)$. This means that the number of control messages is reduced by factor c. The reduction (decrease) of filter forwarding overhead in our filter handling scheme O_f is thus simply,

$$O_f\left(B_1 \rightarrow B_2\right) = c\left(\mathscr{R}_2^{\rightarrow 1}\right). \tag{6.4}$$

This many fewer control messages than in the subscription flooding scheme are forwarded from B_1 to B_2 as the result of one subscription in the sub-network behind B_1.

Forwarding Overhead Trade-Off

In our filter handling scheme, we trade filter forwarding overhead off against notification forwarding overhead. Using the above formulae, we can now describe for one particular edge of the broker network graph the conditions that have to be fulfilled to achieve a benefit with respect to the number of exchanged messages against the subscription flooding scheme.

Our scheme is beneficial with respect to the total number of messages (subscriptions and notifications) sent from a broker B_1 to a broker B_2 if the subscription frequency in the subnetwork behind R_1 times $O_f\left(B_1 \rightarrow B_2\right)$ is larger than the publication frequency of notifications in the same subnetwork times $O_n\left(B_1 \rightarrow B_2\right)$.

[4] Note that this is a subset of $\mathscr{R}_2$'s routing table, and not $\mathscr{R}_1$'s.

6.3.4 Effect Propagation

So far, the evaluation of the filter handling benefits focused on one isolated broker of the broker network. We shall now investigate the propagation of the forwarding overhead effects by regarding one broker and all its neighbor brokers.

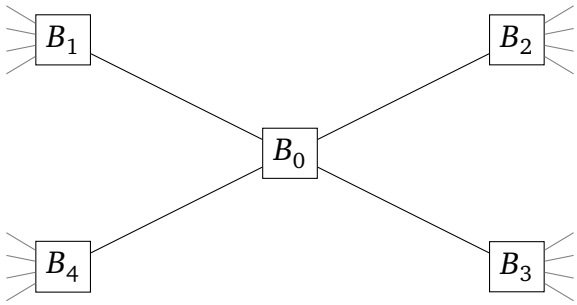


Figure 6.11.: Section of an example broker network

Figure 6.11 shows as an example a broker B_0 with four neighbor brokers. The routing table of B_0 consists of four subsets of routing entries, $\mathscr{R}_0 = \left\{\mathscr{R}_0^{\rightarrow 1}, \mathscr{R}_0^{\rightarrow 2}, \mathscr{R}_0^{\rightarrow 3}, \mathscr{R}_0^{\rightarrow 4}\right\}$. From the subordinate brokers, we only regard the routing table subsets for B_0, i.e., $\mathscr{R}_1^{\rightarrow 0}, \mathscr{R}_2^{\rightarrow 0}, \mathscr{R}_3^{\rightarrow 0}, \mathscr{R}_4^{\rightarrow 0}$.

We describe the propagation of forwarding overhead effects through two characteristic numbers: The number N_{f}^{Δ} describes how many fewer filter handling operations are required in the example network using our filter handling scheme than with the subscription flooding approach, when a single subscription filter is forwarded by one of $B_1, \ldots, B_4$ as the result of a client's subscription in the respective subnetwork "behind" the broker. Similarly, the number N_{n}^{Δ} describes the number of additional notification handling operations throughout the network required in our filter handling scheme because of imperfect filtering, when a notification is received by one of B_0's neighbors out of its respective subnetwork.

Given these two numbers, we can state that the application of our merging-based filter handling scheme pays off with respect to the overall number of messages forwarded between and handled by brokers if the publication frequency of notifications times N_{n}^{Δ} is less than the subscription frequency times N_{f}^{Δ}.

For the following discussion, we abstract from the example in Figure 6.11 and assume that B_0 has N_{b} neighbor brokers.

Filter Forwarding Overhead

In the subscription flooding approach, every new subscription filter that one of B_0's neighbor brokers receives out of its subnetwork results in N_{b} filter handling operations

in the topology section, one for each broker-to-broker link. We denote this number with $N_{\mathrm{f}}^{\mathrm{flooding}}$.

If a subscription filter is to be forwarded from a neighbor broker to B_0, say from B_1, the filter forwarding overhead reduction in our filter handling scheme is $O_{\mathrm{f}}\left(B_1 \rightarrow B_0\right)$. Hence, $1 - O_{\mathrm{f}}\left(B_1 \rightarrow B_0\right)$ subscription filters are sent from B_1 to B_0. The onward filter forwarding reduction from B_0 to its other neighbor brokers B_i is $O_{\mathrm{f}}\left(B_0 \rightarrow B_i\right)$. Hence, the total reduction on these paths is $(1 - O_{\mathrm{f}}\left(B_1 \rightarrow B_0\right)) \cdot (1 - O_{\mathrm{f}}\left(B_0 \rightarrow B_i\right))$. This is the case for each of the neighbors, hence the total number of filter forwarding operations is

$$N_{\mathrm{f}}^{\mathrm{merging}} = \left(1 - O_{\mathrm{f}}\left(B_1 \rightarrow B_0\right)\right) + \sum_{i=2}^{N_{\mathrm{b}}} \left(\left(1 - O_{\mathrm{f}}\left(B_1 \rightarrow B_0\right)\right) \cdot \left(1 - O_{\mathrm{f}}\left(B_0 \rightarrow B_i\right)\right)\right) \quad (6.5)$$

The total number of saved filter forwarding transactions in our filter handling scheme is $N_{\mathrm{f}}^{\Delta} = N_{\mathrm{f}}^{\mathrm{flooding}} - N_{\mathrm{f}}^{\mathrm{merging}}$. This many fewer filter handling operations (forwarding and recepient's processing) are required in total in our filter handling scheme than in the subscription flooding approach.

Notification Forwarding Overhead

Notification forwarding overhead is calculated similarly. In this case, the matching probability m has to be taken into account. For each notification received by one of B_0's neighbors, say B_1, out of its respective subnetwork, the number of notifications forwarded to B_0 in the filter flooding approach is $m\left(\mathscr{R}_1^{\rightarrow 0}\right)$. To each of its other neighbor brokers B_i, B_0 forwards $m\left(\mathscr{R}_1^{\rightarrow 0}\right) \cdot m\left(\mathscr{R}_0^{\rightarrow i}\right)$ notifications. Hence, the total number of notification forwarding operations is

$$N_{\mathrm{n}}^{\mathrm{flooding}} = m\left(\mathscr{R}_1^{\rightarrow 0}\right) + \sum_{i=2}^{N_{\mathrm{b}}} \left(m\left(\mathscr{R}_1^{\rightarrow 0}\right) \cdot m\left(\mathscr{R}_0^{\rightarrow i}\right)\right) \quad (6.6)$$

In our filter handling scheme, $m\left(\mathscr{R}_1^{\rightarrow 0}\right) + O_{\mathrm{n}}\left(B_1 \rightarrow B_0\right)$ notifications are forwarded from B_1 to B_0 and $\left(m\left(\mathscr{R}_1^{\rightarrow 0}\right) + O_{\mathrm{n}}\left(B_1 \rightarrow B_0\right)\right) \cdot \left(m\left(\mathscr{R}_0^{\rightarrow i}\right) + O_{\mathrm{n}}\left(B_0 \rightarrow B_i\right)\right)$ from B_0 to all other neighbor brokers, in total:

$$\begin{aligned} N_{\mathrm{n}}^{\mathrm{merging}} = {} & m\left(\mathscr{R}_1^{\rightarrow 0}\right) + O_{\mathrm{n}}\left(B_1 \rightarrow B_0\right) \\ & + \sum_{i=2}^{N_{\mathrm{b}}} \left(\left(m\left(\mathscr{R}_1^{\rightarrow 0}\right) + O_{\mathrm{n}}\left(B_1 \rightarrow B_0\right)\right) \cdot \left(m\left(\mathscr{R}_0^{\rightarrow i}\right) + O_{\mathrm{n}}\left(B_0 \rightarrow B_i\right)\right)\right). \end{aligned} \quad (6.7)$$

The total number of notifications that are unnecessarily forwarded in our filter handling scheme as the result of one notification publication is $N_{\mathrm{n}}^{\Delta} = N_{\mathrm{n}}^{\mathrm{merging}} - N_{\mathrm{n}}^{\mathrm{flooding}}$.

In the next section, we assume specific filter and notification characteristics for a real-world scenario of aeronautical event notification dissemination, and describe the results of experiments using different numbers of filters and penalty score values.

6.4 Practical Evaluation

To practically evaluate the presented filter handling scheme, we designed a test scenario focused on a day in the European airspace in the year 2022, and created a large set of appropriate scenario data, for which we used as basis real historic flight data provided by EUROCONTROL. Subsection 6.4.1 details the creation and properties of these experimental test data.

The experiments with flight subscriptions and aeronautical notifications aimed to derive statements on the effectivity of the filter handling scheme in different situations. As in the experiments described in the previous chapter, we simulated one broker's processing of subscriptions and notifications. Since it was found in the previous experiments that both the penalty score threshold p_0 and the total number of filters in the original set, $|\mathscr{G}|$, affect the reduction r and the achievable precision p as well as the cover probability c, we varied in our experiments the number of flights for which subscriptions were loaded in the routing table and the threshold. Furthermore, we investigated the effect of filter space density by carrying out the experiments for sets of random flights on one hand and sets compiled exclusively of subscriptions for flights departing from Frankfurt on the other hand. Subection 6.4.2 presents and discusses experimental results.

Subsection 6.4.3 finally takes the experimental results as input to an evaluation of effect propagation on the basis of the theory prepared in the previous section to derive statements on the overall benefit of our filter handling scheme.

6.4.1 Test Data: Aeronautical Subscriptions and Notifications

For our experiments, EUROCONTROL provided us with a set of historic flight plan data from two days in August 2007. The data set includes the flight plan routes of all 58,492 IFR flights that were conducted in European airspace on these days. We used these data as the basis for the flight subscriptions in the target scenario of a day in European airspace in 2022. By extrapolating the real historic flight plan data, we generated realistic future trajectory data. We temporally moved the 4-dimensional flight profiles to the year 2022, doubled the number of flights per day to account for the expected increase in air traffic, and tripled the resolution (number of legs) of the trajectories to

achieve realistic high resolution Business Trajectories. The flight trajectories thus created consist of approximately 70 legs on average. Flight subscriptions were generated by creating a spatiotemporal filter F_i for each leg of the flight. A flight subscription is the set $\mathscr{F} = \{F_i\}$ of all spatiotemporal filters pertaining to the trajectory legs of the flight.

We further generated appropriate event notifications for the experiment by creating spatiotemporal notifications for permanent and temporary aeronautical events, each with an extent in horizontal and vertical space (2-dimensional region and altitude interval). The distribution and extent of the events' temporal and spatial effectivitiy was chosen as realistic as possible with respect to the scenario. The set of synthetic notifications $\mathscr{N}$ was generated thus that the matching probability m of a flight subscription $\mathscr{F}$ is approximately 0.16 %, $m_{\mathscr{N}}(\mathscr{F}) \approx 0.0016$. In our experiments, we published 10,000 event notifications resulting in a flight being affected by 16 events on average.

Appendix C provides detailed explanations on how the experimental subscription and notification data were created.

6.4.2 Experimental Results

In the experimental setup, different sets of flight subscriptions and aeronautical event notifications were processed simulating a broker's filter handling processes. After the processing of a number of subscriptions, the routing table reduction r was calculated, and, using sets of 10,000 event notifications and sets of 50 flight subscriptions, the filtering precision r and the cover probability c were determined.

Two different types of flight subscription sets were used in the experiments. One type of sets contained subscriptions for random flights, thus exhibiting a random distribution of the filters in space and time. In the other type, only subscriptions for flights departing from Frankfurt airport were contained resulting in much higher local density in the horizontal spatial dimensions than in the set with random flight subscriptions.

Reduction and Precision

We first executed the merging experiments with varying values for the penalty score threshold p_0 at fixed input set size. Figure 6.12 shows as an example the achieved precision and reduction at varying threshold values for a set of 500 flight subscriptions from both types of subscription sets. As one would expect, filters from the Frankfurt flight subscriptions are merged at lower threshold values, because the filter set exhibits a higher local density.

Next, we investigated the impact of the number of flight subscriptions by varying this number at a fixed threshold. Figure 6.13 shows as an example the results for a

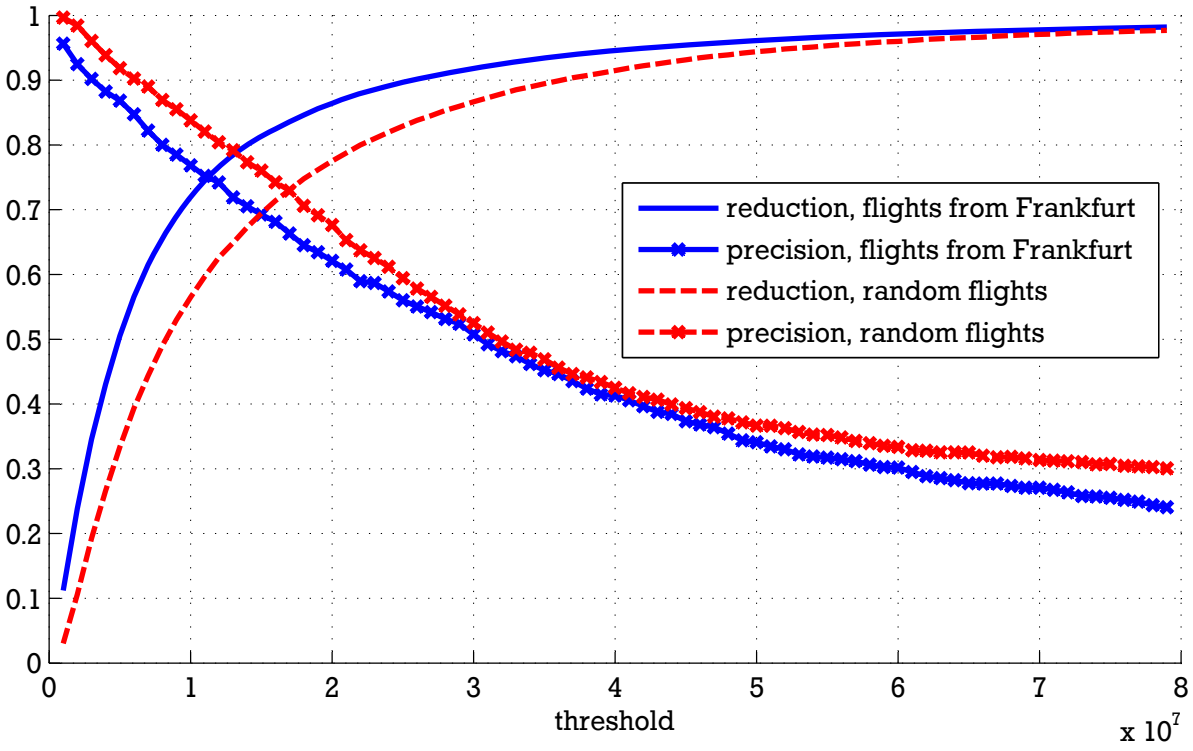


Figure 6.12.: Reduction and precision of reduced sets of 500 flight subscriptions at varying penalty score thresholds

constant threshold value of 10,000,000. Obviously, the number of flight subscriptions does not considerably affect the reduction and precision values at a given penalty score threshold, which is surprising at first glance because it is different from the results we obtained with the two-dimensional synthetic test data. While reduction increases very slowly with the number of subscriptions, it appears that most of the filters representing the individual legs of the randomly chosen flights are too distant to be merged at this threshold value. Probably, the majority of created mergers consists of filters for legs of the same flight. The fraction of legs for which this happens remains (almost) the same for each newly added subscription, hence reduction and precision do not change much with more added flight subscriptions.

This effect is not as strong with the Frankfurt flight subscriptions as with the random ones. For the former, an increasing number of subscriptions yields constantly increasing reduction and decreasing precision, which is again an effect of the higher local density. Spatiotemporal filters for legs from different flights are merged, and as more flight subscriptions are added, the local density of the filter space (in the horizontal spatial dimension, around the location of Frankfurt airport) further increases.

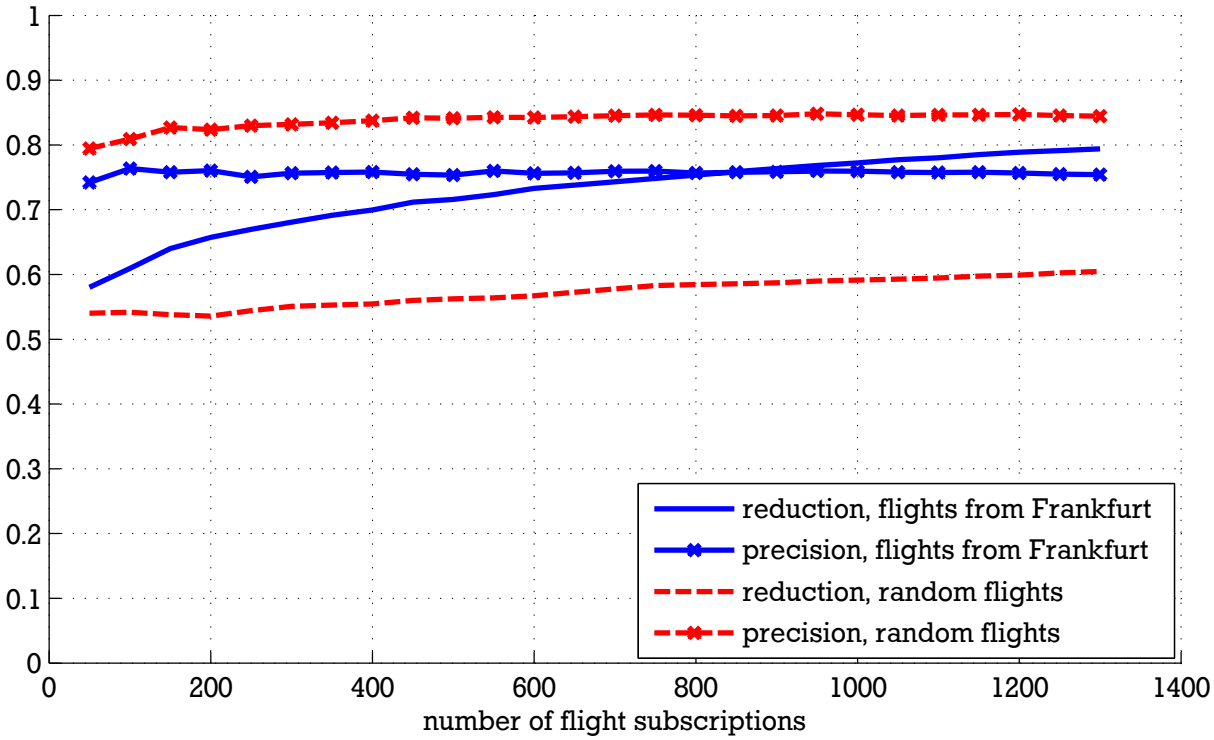


Figure 6.13.: Reduction and precision of reduced sets of flight subscriptions of different sizes at fixed penalty score threshold $p_0 = 10^7$.

Forwarding Overheads

In order to evaluate the forwarding overhead trade-off, we determined in the same experimental setup as before the cover probability c (which equals the filter forwarding overhead reduction O_f), the matching probability m, and based on the latter, the notification forwarding overhead O_n. c was determined for each processed set $\mathscr{G}$ by taking a number of (not previously added) flight subscriptions of the same type (random or from Frankfurt) as the subscriptions in the original set $\mathscr{F}$ and determining the fraction of filters that is covered by some filter from $\mathscr{G}$.

Figure 6.14 shows the increase/decrease of the forwarding overheads, O_f and O_n, for sets of 500 flight subscriptions at varying thresholds. In this comparison, the two types of sets produce widely different results. Whereas the filter forwarding overhead reduction is much larger than the notification forwarding overhead increase for the Frankfurt subscriptions at low threshold values already, and constantly increases, the processed set of random flight subscriptions produces more unnecessarily forwarded notifications than it saves forwarded subscription filters at low thresholds. However, O_f grows faster with the threshold value, and is eventually larger than O_n.

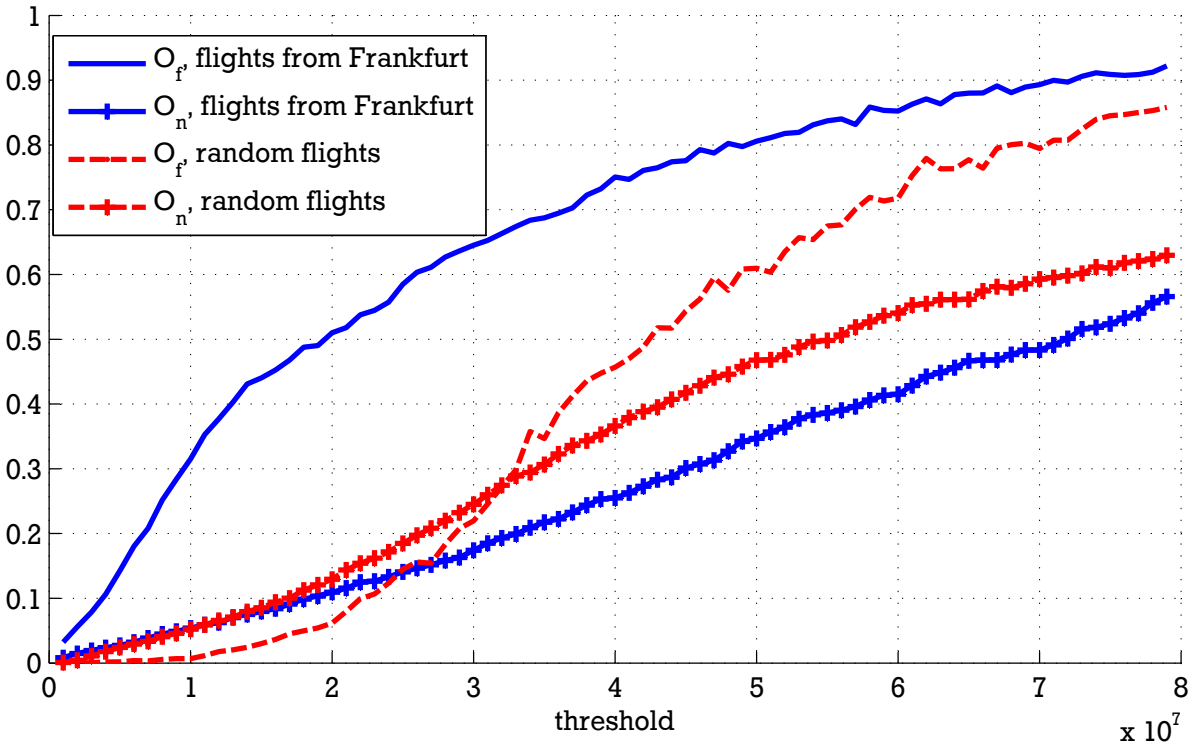


Figure 6.14.: Filter and notification forwarding overhead for sets of 500 flight subscriptions in dependency of penalty score threshold

Forwarding Overhead Trade-off

In order to be able to evaluate the forwarding overhead trade-off, we put the filter forwarding overhead reduction O_f in relation with the notification forwarding overhead O_n, and compare the *forwarding overhead ratio* O_f/O_n for the different flight subscription set types at varying set sizes and threshold values.

Figure 6.15 shows the forwarding overhead ratio for different filter set sizes for both types of subscription filters. The random flight subscriptions generally produce very poor results. The ratio is constantly increasing but seems to approximate a maximum of not more than 1.5 for very high threshold values. The Frankfurt flight subscriptions produce much better forwarding overhead ratio. The curve is not constantly increasing, but reaches a maximum, and constantly decreases from there, seemingly also approximating 1.5.

Varying the number of flight subscriptions at constant threshold (Figure 6.16), we found that the forwarding overhead ratio generally increases with increasing numbers of subscriptions for both types of sets and all threshold values. However, the ratio value increases much steeper for the Frankfurt flight subscription sets and generally reaches a much higher level.

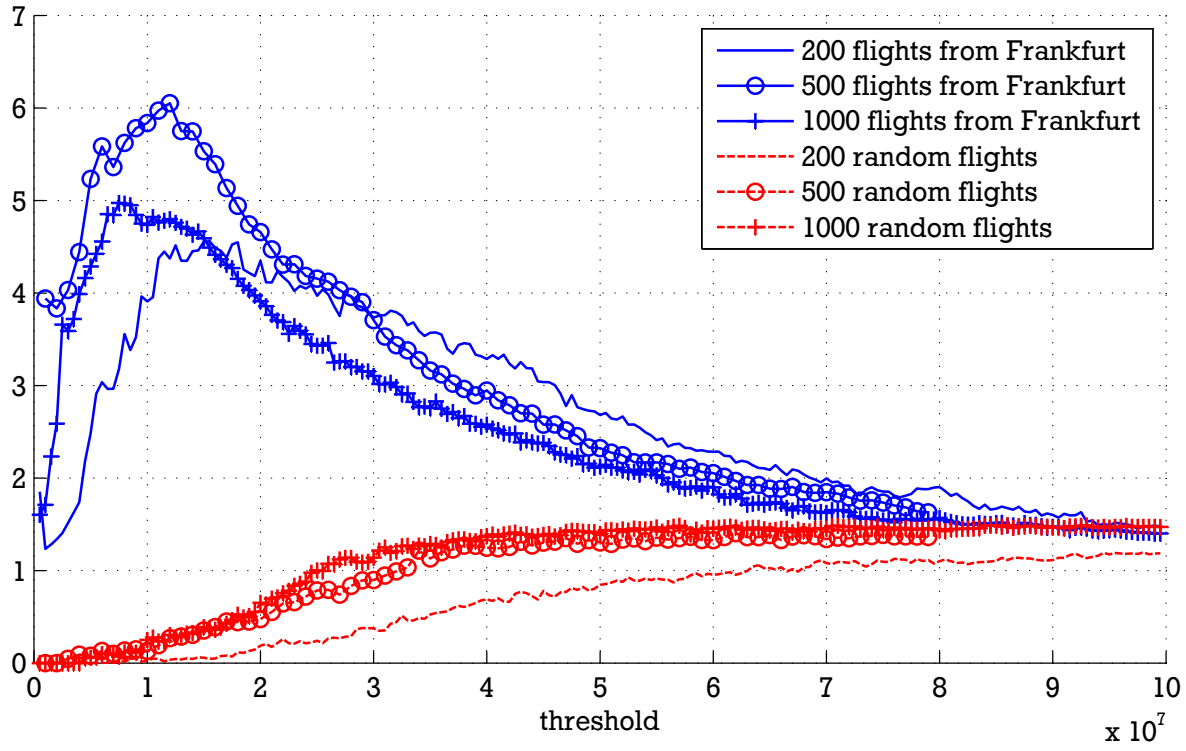


Figure 6.15.: Forwarding overhead ratio O_f/O_n for different sets of flight subscriptions in dependency of penalty score threshold

6.4.3 Evaluation of System-Wide Effects

In this subsection, the propagation of the filter handling scheme effects shall be evaluated based on the experimental results to derive statements on the overall benefit of our filter handling scheme in the given scenario.

To do so, we have to assume a particular broker network topology and client behavior. In the following, we first discuss the effects with respect to a particular example, and shall come back to other setups at the end of this section.

We assume a root broker and four subordinate brokers, each with an arbitrarily structured subnetwork behind, and investigate the effects for this setup. Figure 6.11 above[5] shows such a setup. We assume that each of the subordinate brokers has already forwarded 500 flight subscriptions to B_0, hence $\left|\mathscr{R}_0^{\rightarrow 1}\right| = \left|\mathscr{R}_0^{\rightarrow 2}\right| = \cdots = 500$ and $\left|\mathscr{R}_1^{\rightarrow 0}\right| = \left|\mathscr{R}_2^{\rightarrow 0}\right| = \cdots = 1500$ (with filter flooding). Let us further assume some subscription similarity among the active subscriptions in one subnetwork. This would be the case if, e.g., the subscriber clients in the subnetwork behind B_1 subscribe only for flights departing from Frankfurt, those behind B_2 for flights from Paris, and so on. Let us further assume for the sake of simplicity that the similarity among the subscriptions

[5] Section 6.3.4, Page 154

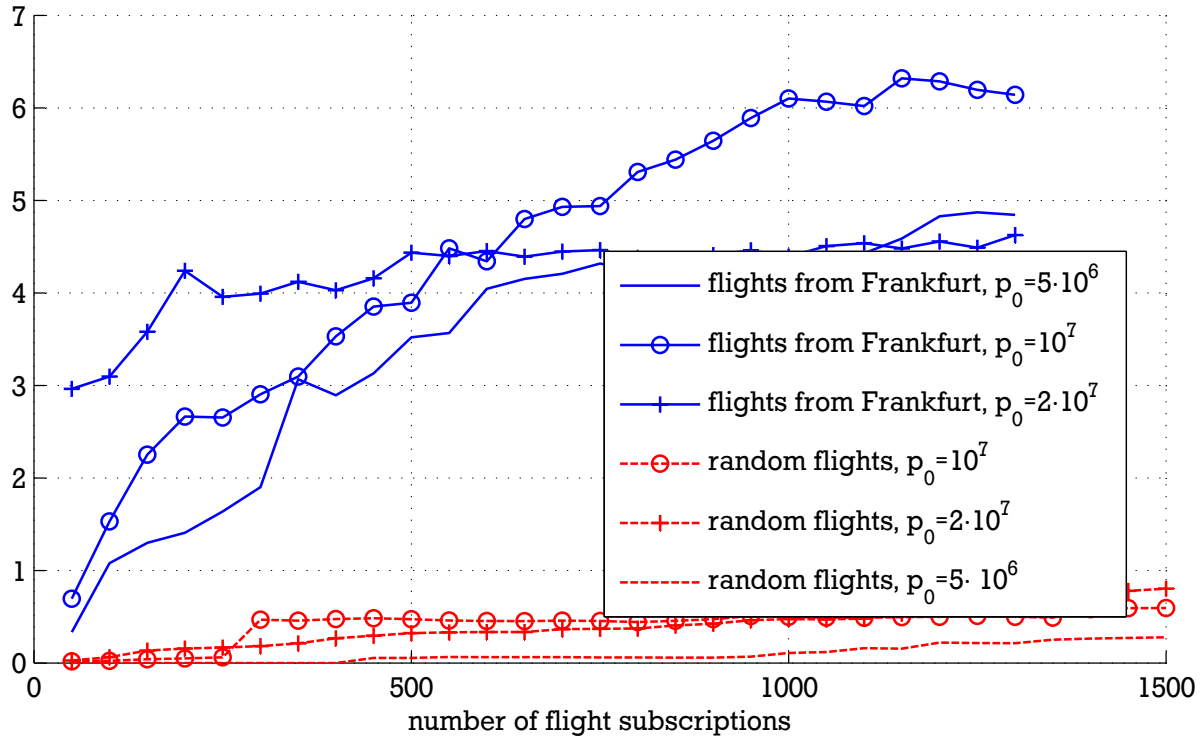


Figure 6.16.: Forwarding overhead ratio $O_\mathrm{f}/O_\mathrm{n}$ for sets of flight subscriptions of different sizes at constant penalty score threshold values

Table 6.1.: Experimental results for two sets of flight subscriptions at penalty score threshold $p_0 = 10^7$

characteristic	$\mathscr{R}_0^{\rightarrow i}$: 500 Frankfurt flights	$\mathscr{R}_i^{\rightarrow 0}$: 1500 random flights
r	0.7159	0.6119
p	0.7534	0.8441
m	0.1759	0.4094
$O_\mathrm{f} = c$	0.2289	0.0444
O_n	0.0588	0.0748

in each subnetwork is equal. Then we can assume the same characteristics (reduction, precision, matching probability, cover probability, and forwarding overheads) for each of $\mathscr{R}_0^{\rightarrow 1}, \mathscr{R}_0^{\rightarrow 2}, \ldots$, for which we take the results obtained in our experiments for the Frankfurt flight subscription sets. For the sets $\mathscr{R}_0^{\rightarrow 1}, \mathscr{R}_0^{\rightarrow 2}, \ldots$, we assume less similarity because the subscriptions have mixed, and take as characteristics of the sets the results obtained in our experiments for sets of random flight subscriptions.

If we apply our merging-based filter handling scheme with a penalty score threshold of $p_0 = 10^7$, Table 6.1 shows the characteristics of the subscription sets in this example.

The system-wide reduction of the routing table subsets can be derived directly from the values in the table. For the subsets of $\mathscr{R}_0$, it is $r\left(\mathscr{R}_0^{\rightarrow i}\right) = 0.6119$. For the four routing table subsets $\mathscr{R}_i^{\rightarrow 0}$, it is $r\left(\mathscr{R}_i^{\rightarrow 0}\right) = 0.7159$, which gives an overall mean reduction of $\overline{r} = \left(4 \cdot r\left(\mathscr{R}_0^{\rightarrow i}\right) + 4 \cdot r\left(\mathscr{R}_i^{\rightarrow 0}\right)\right)/8 = 0.6639$.

Regarding filter forwarding overhead reduction, the values $O_\mathrm{f}\left(B_0 \rightarrow B_i\right) = 0.0444$ and $O_\mathrm{f}\left(B_i \rightarrow B_0\right) = 0.2289$ from Table 6.1 are used[6] in Equation (6.5), which gives $N_\mathrm{f}^{\mathrm{merging}} = 2.9817$. With $N_\mathrm{f}^{\mathrm{flooding}} = 4$, we get $N_\mathrm{f}^{\Delta} = 1.0183$.

Equation (6.6) and (6.7) give the notification forwarding numbers $N_\mathrm{n}^{\mathrm{flooding}} = 0.6254$ and $N_\mathrm{n}^{\mathrm{merging}} = 0.8251$, which gives in total $N_\mathrm{n}^{\Delta} = 0.1997$.

This means that in the above example, the application of our merging-based filter handling scheme pays off with respect to the overall number of messages forwarded and handled by brokers in this broker hierarchy level if the publication frequency of notifications is not more than the forwarding overhead ratio

$$\frac{N_\mathrm{f}^{\Delta}}{N_\mathrm{n}^{\Delta}} = 5.099$$

times the subscription frequency.

We calculated the forwarding overhead ratio also for three other example scenarios, changing in each the characteristics of the setup that affect the forwarding overheads and thus the overall result:

- Assuming a different penalty score threshold. Changing the threshold to $p_0 = 5 \cdot 10^6$ in the above example results in the values: $\overline{r} = 0.4418$, $N_\mathrm{f}^{\Delta} = 0.4838$, and $N_\mathrm{n}^{\Delta} = 0.0883$, which gives a forwarding overhead ratio of 5.479. For a threshold value of $p_0 = 2 \cdot 10^7$, the overall routing table reduction is $\overline{r} = 0.845$ and the forwarding overhead ratio is 4.7442.

  Hence, in this example setup, the lower threshold value $p_0 = 5 \cdot 10^6$ is the best choice, although the choice of the threshold value does not have a big impact on

[6] Note that $O_\mathrm{f}\left(B_0 \rightarrow B_i\right)$ is derived from $R_i^{\rightarrow 0}$ and $O_\mathrm{f}\left(B_i \rightarrow B_0\right)$ is derived from $R_0^{\rightarrow i}$, see footnote in Subsection "Filter Forwarding Overhead" in Section 6.3.3.

the forwarding overhead ratio, which is surprising considering that widely different overall routing table size reductions are achieved.

- Assuming a different topology. We changed the number of subordinate brokers in the example topology to three (resulting in $N_f^\Delta/N_n^\Delta = 5.2547$) and to five ($N_f^\Delta/N_n^\Delta = 5.6262$) at $p_0 = 5 \cdot 10^6$. The overall routing table size reduction is the same as above $\overline{r} = 0.4418$.

  Hence, the number of subordinate brokers also seems not to impact the result considerably.

- Assuming less similarity. With all routing table subsets consisting of subscriptions for random flights, the resulting overall routing table size reduction was 0.3512, and the forwarding overhead ratio was 0.3546 at $p_0 = 5 \cdot 10^6$, $\overline{r} = 0.5872$ and $N_f^\Delta/N_n^\Delta = 1.0773$ at $p_0 = 10^7$, and $\overline{r} = 0.7903$ and $N_n^\Delta/N_f^\Delta = 1.0544$ at $p_0 = 2 \cdot 10^7$, assuming four subordinate brokers.

  Obviously, the forwarding overhead ratio is very low compared to the setups with filter sets with similarity. It is better for high threshold values, which was found in the experimental results in the previous section already. As can be seen in Figure 6.15, a forwarding overhead ratio of more than 1.5 can however not be achieved.

A central conclusion we can draw from the experimental results and their evaluation with respect to effect propagation throughout the broker network is that similarity among subscriptions remains an important driver for the effectiveness of the imperfect merging approach. Only the Frankfurt flight subscription sets can be merged under particular conditions such that the forwarding overhead ratio is very beneficial. The subscription sets with less similar filters (those with random flight subscriptions) seem to reach at most a forwarding overhead ratio value of 1.5. This means that the publication frequency may at most be 1.5 times the subscription frequency in the respective subnetworks.

Hence, whereas the routing table size reduction as a beneficial effect of our filter handling scheme is prevalent in all investigated setups, a benefit with respect to the total number of messages exchanged between brokers can only be achieved under the assumption that either the subscriptions issued by clients in some subnetwork exhibit some similarity, or the publication frequency is not much higher than the subscription frequency.

7 Conclusions and Future Work

In this thesis, several developments are presented as elements of a SWIM publish/subscribe notification service for aeronautical events in the future ATM system: a spatial and a temporal model for aeronautical data, a spatiotemporal subscription model based on two types of filters, (geo-)spatial and interval filters, different merging strategies for these filters, and finally the application of the filter merging approach in the filter handling processes in a distributed publish/subscribe system. This chapter concludes this thesis with a summary of the achieved results and findings, based on which we present ideas for future work. Section 7.1 revisits the spatiotemporal aeronautical event model. In Section 7.2, the findings pertaining to the spatiotemporal filter model are briefly summarized, and ideas are presented how to further exploit the foundation provided by the model for other application areas. Section 7.3 finally looks at the filter merging approach and the resulting filter handling scheme for distributed publish/subscribe brokers again and briefly discusses related topics that may enhance the overall approach to make it more beneficial for specific applications.

7.1 Aeronautical Event Notification Service

Chapter 3 presented the analysis of requirements of a SWIM notification service with respect to a model of aeronautical events. For the development of this model, we focused on the spatial and temporal aspects of events to allow for the subscription to these aspects based on a SWIM User's interest, e.g., the 4-dimensional trajectory of a flight.

The model is based on a model of states and events of features. States describe the condition of a feature over a period of time, a temporal interval, through the assignment of specific values to the feature's modeled properties (attributes). Events describe the state changes, i.e., the transition from one state to the other, through a value change of one or more feature attributes. We argued that this model of states and events is complete in the sense that the past, present, and future of (happenings in and conditions of) the modeled world can be sufficiently described in this model.

However, being a model of the valid time dimension only, this model will not be sufficient for applications that employ the notion of planned, future events, for which notifications must be disseminated in advance of the event's valid time, like aeronautical events. For obvious reasons, information distribution and processing cannot be assumed to happen in real-time. Instead, these steps in AIS and AIM operations will always require a temporal model of information to account for different distribution

paths with varying latency (from seconds to weeks). This made it necessary to extend the minimum model of feature states and events with the notion of two prioritized layers of information, a baseline and a temporary layer, with permanent and temporary events, respectively.

We find that this model is complete and sufficient to provide for the implementation of any real-world scenario through the flexibility introduced by the distinction of baseline and temporary conditions, which is arbitrary and can therefore be defined based on application needs.

7.1.1 Issue: Transient Event Notifications

As became clear in the discussion of real-world application requirements resulting in the introduction of baseline and temporary feature states, application-specific times (such as latency leading to different subscriber notification times), albeit not impacting the conceptual model of valid time, must be taken into account to allow for the temporality model's implementation in real-world applications.

With our model of advance notifications of events, the following issue pertaining to client subscription time can be identified: Assume that a notification for an event with valid time t_e is published at publication time t_p before t_e, $t_p < t_e$. If a client subscribes at time t_s after t_p but before t_e to a temporal interval that contains t_e, $t_p < t_s < t_e$, this client might never be notified of the event, because the event notification has been delivered to all subscribed clients already.[1] This problem is due to the transient nature of event notifications in the publish/subscribe system. Once a notification has been routed through the network and delivered to the consumers, it ceases to exist in the system.

This is also an issue for practical situations in real-world applications when refraining from idealized assumptions with respect to client connectivity. M. Cilia *et al.* [37] and P. Guerrero in his Master's thesis [89] discussed notification transience as an issue for mobile publish/subscribe clients, which may be temporarily disconnected and/or change the access point to the system (the border broker). They proposed a notification buffer component of a border broker. The buffer component has producer semantics in that it independently replays event notifications upon a client's reconnect (and resubscription). In a similar approach, G. Li *et al.* [122] proposed an independent component of a publish/subscribe system that acts as a notification sink by subscribing to all notifications and centrally storing all published notifications in a database. Upon a new client subscription, the past notifications are replayed. While this approach suffers from

[1] For simplicity, we disregard the latency in this discussion, and assume that the publication time and the notification time at any client are the same.

single-point-of-failure concerns, both proposed solutions can alleviate the problem for our application scenario as well. As a further advantage of our event model in this setup, the approaches would not suffer from the intrinsic scalability problem of notification buffer size: In traditional event dissemination scenarios, the question would always be for how long or what number of notifications are kept in the buffer. Storing all notifications forever is obviously a severe scalability limitation. With notifications published in advance of the event valid time, every notification would only have to be stored until the event's valid time, thus naturally bounding notification buffer size.

7.1.2 Future Work: Integrated Pull- and Push-Semantics

We shall propose an alternative approach to the notification transience issue based on the conceptual temporality model of states and events. We argue that an integration of different communication patterns would be a better solution, and sketch in the following the semantics of a data distribution system for this purpose.

In the SWIM context, we can assume that there is one global valid time state of the aeronautical environment at any moment in time. The state information of aeronautical features may be distributed, but this distribution is free of overlap such that any aeronautical feature is governed by exactly one Producer stakeholder, which maintains the state information. In this global information space, a User client initially connecting to the information distribution system of SWIM actually needs to update its internal information about the state of the aeronautical environment by getting the currently valid state of all features (a snapshot of the world) it is interested in plus all previously published notifications for future events affecting its part of interest of the global information space.

This bootstrap action is obviously much better handled in a data pull pattern like request-reply as opposed to the data push-based publish/subscribe pattern. To maintain the decoupled nature of Producer and Consumer clients, this bootstrap action could be handled in an anonymous requst-reply fashion [141], where the information distribution system mediates the User request to the appropriate Providers based on the requested information items. This assumes that there is global knowledge of which Provider maintains which part of the information space, which could be achieved in a way similar to advertisements, a well-studied concept in publish/subscribe systems [28, 137].

As a simple example, assume that the responsibility of the information space is distributed spatially, i.e., Providers of information are responsible for all features located in mutually exclusive parts on or over the Earth's surface. This is already the case in today's AIS. It would be then sufficient to maintain advertisements for these areas in

the broker network to allow for every broker to decide in which direction to forward (possibly decomposed parts of) a User's bootstrap request. For the reply path to the User, the brokers that forwarded the request would install a dedicated channel only for this request that is created upon forwarding a request and ceased upon forwarding the reply. This way it is guaranteed that only the requesting User receives the reply to its individual request.

If the User includes in the request the time of the last update of its information base (e.g., the moment when it previously disconnected from the system), it is not even necessary to send to the User the full baseline information (all feature states) but only all changes that occured between that time and now. This change information can simply take the form of event notifications.

After the bootstrap phase, the User's information base is up-to-date, and the User can start receiving event notifications published from this moment on through the system's regular publish/subscribe semantics. By integrating the pull- and the push semantics in the registration phase (request and subscription), the User must only once communicate the part of the information space it is interested in. The information contained in the request already contains the subscription information. A subscription could be stored in the broker's routing table and forwarded appropriately in an atomic step with processing the request.

Given appropriate algorithms for a broker's handling of the request and reply, this approach could solve a known problem of distributed publish/subscribe systems, namely that network latency makes it impossible to guarantee that a subscriber receives a notification matching a previously registered subscription if the time between subscription and publication is very small. In that case it can happen that the subscription filter has not yet been processed by all brokers on the way between producer and consumer in the moment of publication. The same problem exists with advertisements and subscriptions, where the latter may not be forwarded correctly if a previously registered advertisement has not yet been processed by all necessary brokers. If the push and the pull semantics are integrated as sketched above, such an effect would be avoided and it may be possible to guarantee that a matching notification is delivered in any case if the request/subscription time is before the publication time.

7.2 Spatiotemporal Filter Model

The filter model presented in Chapter 4 defines spatial filters and temporal interval filters, which are combined to spatiotemporal filters. In combination with the spatiotemporal event model it allows to subscribe for an event's spatiotemporal effectivity and thus provides the means to describe for event characteristics that are independent of

the specific data domain in which events and notification content is represented, which makes it a versatile model easily applicable beyond the scope of aeronautical information or the SWIM scenario. The formal foundation based on the established frameworks of Allen's operators and the 4-intersection matrix further adds to it being a sound model.

7.2.1 Future Work: Relationship Types

While the geometry filters as designed here employ topological spatial relationships theory, the presented filter relationships formalization lays the foundation for the implementation of other spatial relation filters, be it orientation relationships ("north-of…"), distance relationships ("close", "in 5 m distance") or other topological relationships ("touches"). The same applies to temporal interval filters, where more complicated filter expressions, topological ones based on Allen's operators, or metric ones ("exactly 5 minutes before"), or a combination ("at most 10 minutes after"), could be useful for some applications. They should be easily implementable in the presented framework. However, the impact on the filter merging approach would have to be investigated for each relationship individually.

7.2.2 Future Work: Spatial Representations

We used geometric representations (point, line, polygon) of spatial constructs and, for filter expressions, polygons as representations of regions, as it is common in Geographic Information Systems. This approach comes with extensive computational effort, and the study of efficient geometric algorithms in Computational Geometry is a discipline in its own right. Another way to represent spatiality is through named places. Every spatial construct can be given an identification, and many aeronautical features readily have a name. The relationships between named geographic features could be determined in advance and represented, e.g., in data structures that describe for any two spatial features their topological relation (e.g., "airway 'U123' overlaps airspace 'UIR Rhein' "; "airspace 'UIR Rhein' inside 'Germany' "). While the exclusive use of this approach would probably considerably limit the expressiveness of the spatial filter model, a combination of named features and geometrically represented ones could be employed, which maintains the expressiveness of the geometries but provides efficiency enhancements for named features. U. Leonhardt presented in his Ph.D. work [121] such a model ("semi-symbolic hierarchical location model") to represent and relate locations. Other representations of space and location are required when uncertainty is to be represented, be it the uncertain exact location or extent of a feature or the geometric representation of a ("vague") placename. Fuzzy sets [154], point density

surfaces [110], and other more specific models [40] have been proposed as representations of named places, and the general vagueness of place names has been a well researched topic in Geographic Information Science in recent years [94], see also the author's contribution to this research topic [88].

The usability and advantages of such spatial represenations as filter models for a distributed publish/subscribe system and its feasibility for the filter merging approach would be a promising research direction.

7.3 Filter Merging and Filter Handling Scheme

The size-based and distance-based filter merger quality estimations developed and evaluated in Chapter 5 proved to be simply applicable means to trade off filter precision with reduction of a set of filters. The presented approach allows to adjust this trade-off by a single numerical parameter, the merging penalty score threshold. Experimental results showed that the overall approach worked well, and allowed to compare the four proposed implementations of merger quality estimator functions, out of which the one based on normalized filter distance performed best with respect to achieved filtering quality.

The designed algorithms for the application of the filter merging approach in the filter handling operations of a broker in a distributed publish/subscribe system presented in Chapter 6 were implemented in the publish/subscribe prototype REBECA and worked as expected. The achievable benefits compared to simpler filter handling schemes, however, depend largely on the particular environment of the publish/subscribe system (broker network topology, subscription and notification characteristics), and a benefit for the trade-off between filter forwarding overhead and notification forwarding overhead could be shown for the aeronautical scenario with realistic flight trajectories as subscriptions and synthetic aeronautical events as notifications only under specific assumptions pertaining to the similarity of subscriptions in sub-networks.

7.3.1 Future Work: Dynamic Adaptation of Filter Handling

We assumed one global merging score threshold for the entire system such that merging decisions are made by every broker the same way. A promising extension of the approach (albeit more complicated) is to apply broker-specific merging score thresholds. Since the "best" choice for the threshold value depends largely on the characteristics of the subscriptions and notifications a broker processes, these characteristics could be evaluated online, e.g., in a sliding window fashion for the last x processed subscriptions and the threshold could be adjusted accordingly, to adapt the merging

score threshold. The online evaluation could even dynamically trigger a change of filter handling schemes, e.g., from merging-based strategies to the simpler filter flooding strategy.

In the same way, notification characteristics could be evaluated online (and not, as in our experiments, evaluated in advance) and based on their mean size and mean distance, the filter space normalization could be adjusted dynamically. The feasibily of this approach, however, is questionable, because an efficient evaluation of merging candidates requires —at least in our implementation— a previously determined position of filters in the normalized filter space for the distance calculation. However, appropriate experiments would provide more insight.

7.3.2 Future Work: Notification Matching Using Indexes

Advanced filter handling schemes like the one based on filter merging presented in this thesis serve primarily to achieve higher scalability of distributed publish/subscribe systems. The achieved reduction of the numbers of filters to evaluate for notification matching has however also an obvious positive impact on the efficiency of this task. Many other approaches for efficient filter handling schemes and notification matching have been proposed elsewhere such as S. Bittner's and A. Hinze's subscription tree pruning [23, 21] and S. Bianchi *et al.*'s distributed R-tree [19]. Maintaining an index data structure for subscription filters is an obvious approach to expedite notification matching, and an R-tree is a natural choice for spatiotemporal data as used in our application. The maintenance of such an additional structure is costly and must be carefully balanced with the achieved performance gains. It can be expected that the achievable benefit depends once again on the characteristics of the filters and notifications.

T. Penev proposed in his Master's thesis [146] the application of a Hilbert R-tree [113]. In this approach, the Hilbert space-filling curve is used to assign an individual number value to each filter. This "Hilbert value" is used for ordering the filters in the R-tree. An integration of this approach with the normalized filter space position presented in this thesis seems promising. Since a natural property of Hilbert values is that the distance of points in space relates directly to the distance of the Hilbert values representing these points, the filter distance criterion used for merging decisions could be redefined based on the Hilbert value difference. An in-depth analysis of the required algorithms for reorganizing the R-tree in the case of subscriptions, unsubscriptions, and filter merging with respect to efficiency, and the relation to the achievable efficiency gains for notification matching, would have to be the main targets of investigation.

As a final note, the application of an index structure for subscription filters has a considerable impact on the achievable benefit of our filter handling scheme if computa-

tional effort is taken into account. Our investigation focused exclusively on the number of subscription filters and notifications to forward and handle resulting in the forwarding overhead ratio characteristic. If the computational effort required for processing control messages and matching notifications is taken into account, the achievable overall benefits with respect to efficiency could be much larger. An index structure allows for very efficient notification matching, but requires costly update operations in the case of processing subscriptions or unsubscriptions (such as tree reorganizations). The ratio of the computational effort required for routing table updates and for notification matching would then have to be multiplied with our forwarding overhead ratio . For instance, if filter processing takes t_f time on average and notification matching takes t_n time, our filter handling scheme would then be beneficial with respect to overall efficiency if $t_f \cdot N_f^{\Delta}$ times the subscription frequency is more then $t_n \cdot N_n^{\Delta}$ times the publication frequency, which obviously can result in a completely different conclusion about the benefit of our filter handling scheme.

Appendix

A Implementation

The implementations that have been carried out as part of this thesis are twofold: On one hand, the REBECA framework was extended to support spatial and temporal filters as well as the filter handling processes based on imperfect filter merging (Section A.1). On the other hand, frameworks for interval and spatial filters were implemented (Section A.2). While we used an established open source library for the implementation of spatial filters and required algorithms, our interval logic based on discrete spaces required own implementation work.

This appendix briefly outlines our implementations and presents a selection of key challenges and implemented solutions as examples. Where Java implementation code is cited in the following, error and exception handling that does not serve the discussed functionality has been removed for the sake of readability.

A.1 The REBECA Framework

REBECA is a research implementation of a content-based, distributed pub/sub system that features a modular filter and routing framework that allowed us to implement our filter types and filter handling scheme within the framework without affecting REBECA's core or other modules like the communication subsystem. The different filter merging strategies were also implemented in REBECA.

REBECA was originally partly implemented in the frame of the doctoral dissertation of G. Mühl [134] and has undergone frequent extensions since.

The REBECA framework consists of different modules, each implementing a particular aspect of the distributed pub/sub system. The following subsections explain the modules for brokers and routers, the routing framework, and different filter and content models in more detail, providing the basis for the descriptions of the extensions that were implemented as part of this thesis.

A.1.1 Brokers and Routers

The nodes of the distributed notification service that process and forward subscriptions and notifications, which we previously called "inner brokers", are implemented by the `EventRouter` class in REBECA (Figure A.1). `EventRouters` are independent components

that can run on different computers connected over some communication network[1] and maintain connections with other EventRouters and local brokers.

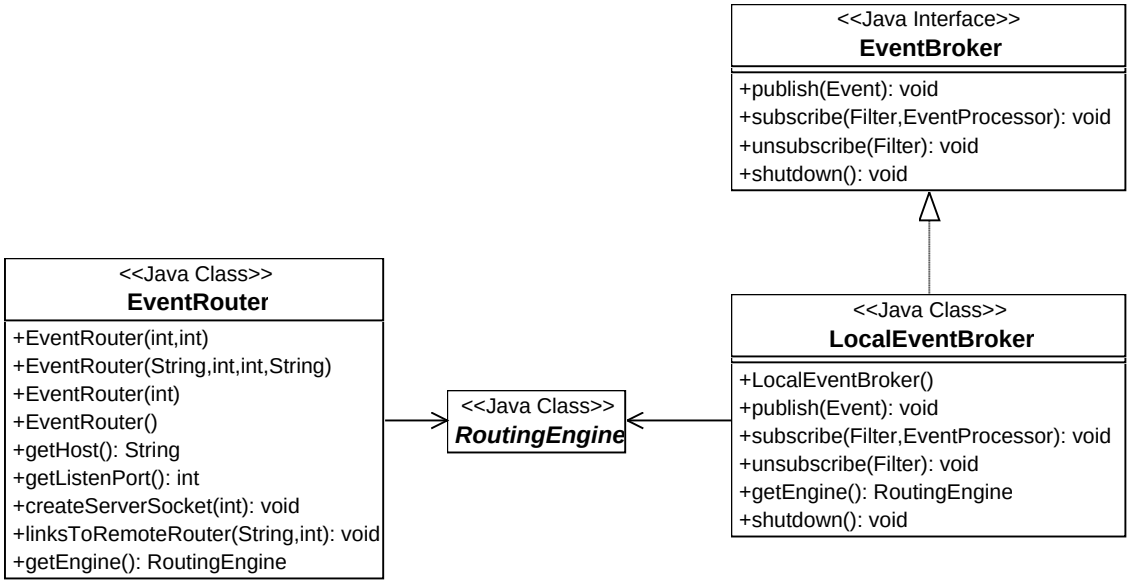


Figure A.1.: REBECA broker classes

Local brokers, which provide the link between a (consumer or producer) client and the notification service, are implemented in class LocalEventBroker. The pub/sub interface with its methods publish, subscribe, and unsubscribe, is defined in the Java interface EventBroker. LocalEventBroker implements this interface by providing the core pub/sub functionality as described in Section 6.1.

The EventRouter class just like the LocalBroker class however merely represent the respective system components. Their main functionality, the actual routing of notifications and handling of subscriptions and unsubscriptions, is delegated to specializations of a fundamental routing class RoutingEngine. Specializations of this class implement the different filter handling optimization approaches like equality-based and covering-based routing as well as the respective processes based on filter merging (called merging-based routing in G. Mühl's early approach).

A.1.2 Routing Framework

The specializations of RoutingEngine that implement the different filter handling optimizations build up a hierarchy of inhereting classes (Figure A.2). These specializations can be used in REBECA interchangeably by external configuration, i.e., which one of the

[1] In fact, only TCP/IP networks are supported by Rebeca. The communication subsystem could however be easily extended such that EventRouters could use other network protocols to communicate.

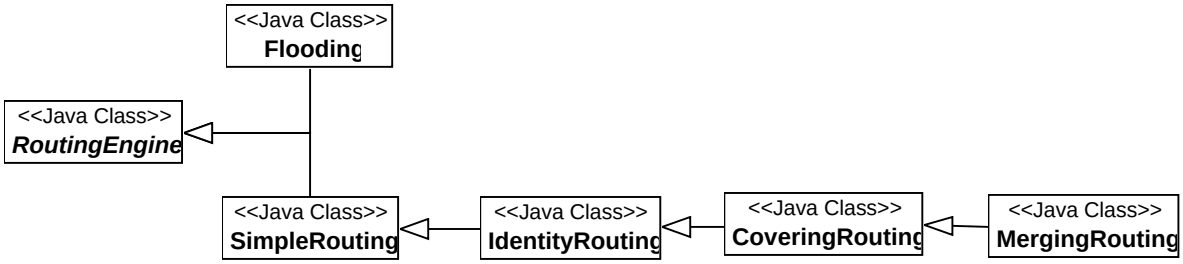


Figure A.2.: REBECA routing framework classes

filter handling approaches is applied in the running system is configured for the whole system at startup. The role of an instance of a `RoutingEngine` or its specializations is the processing and forwarding of incoming control messages and notifications, which take the form of instances of the general class `Event`.

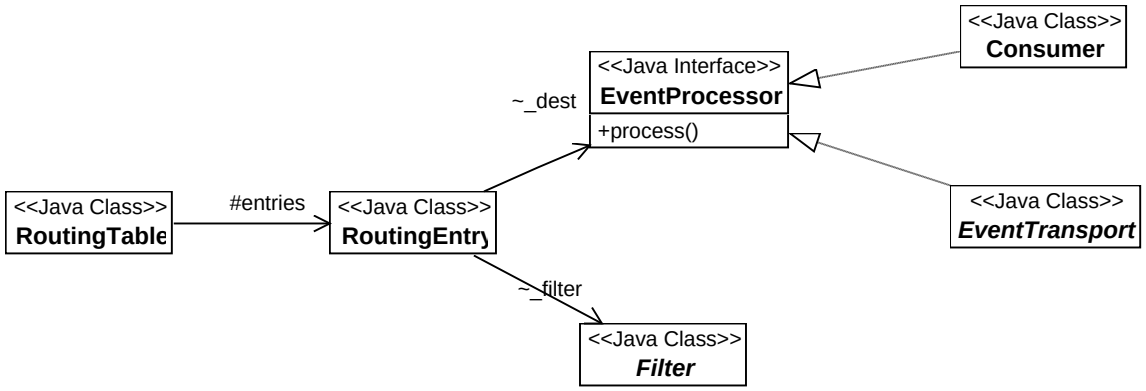


Figure A.3.: REBECA `RoutingTable` and `RoutingEntry` classes

This is achieved using an instance of the class `RoutingTable` (Figure A.3). This class implements the routing table $\mathscr{R}$ of a broker. It maintains a list of `RoutingEntrys`, each consisting of a `Filter` and a destination. Destinations are implemented by the `EventProcessor` interface. It defines one single method, `process`, for events. This method is used to implement the call-back method *notify* for consumer clients, but also to pass events to the communication subsystem to be forwarded to a neighbor broker. This functionality is implemented in the class `EventTransport`.

A.1.3 Filters and Events

The abstract base classes `Filter` and `Event` define the main functionality and operations of (subscription) filters and event notifications (Figure A.4). Broker control messages take the form of `AdminEvents` in REBECA, and are generally handled like regular

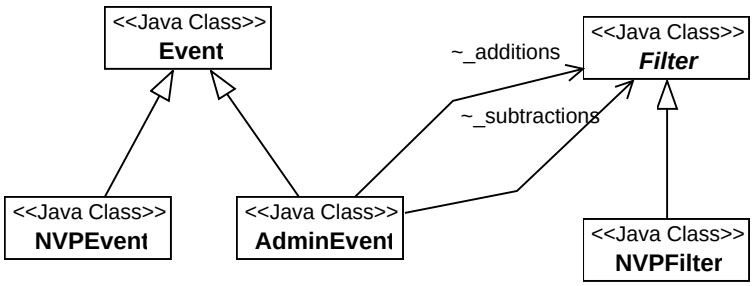


Figure A.4.: REBECA `Filter` and `Event` base classes and name-value pair model realizations as well as the class `AdminEvent` that implements broker control messages

event notifications in the network. An `EventRouter` decides at processing time of an incoming event message how to handle this event. The filter handling scheme operations (routing table update and filter forwarding) are triggered by the reception of an `AdminEvent`.

A.2 Spatiotemporal Filters and Notifications

Our specific filter types are implemented in the classes `IntervalFilter`, `SpatialFilter`, and `SpatiotemporalFilter` (Figure A.5). These classes inherit from `Filter` and overwrite the methods that implement specific filter relationships such as equality, cover, or intersection as well as methods for merging quality estimation and the merging operation itself. To efficiently evaluate the size-based and distance-based merging quality estimation, a filter's size and its centroid's position in the filter space are stored in member variables.

Merger filters are also implemented in the Filter class. A Java collection structure (`ArrayList`) stores all constituents of a merger, and the member method `isMerger` simply checks whether this structure is empty.

Interval, spatial, and spatiotemporal notifications are implemented similarly (Figure A.6). The classes `IntervalEvent`, `SpatialEvent`, and `SpatiotemporalEvent` inherit from the base class `Event`, encapsulate the event information and make available the event characteristics required for matching through *getter* and *setter* methods.

A.2.1 Regions

For the representation of spatial objects, we used the open source Java library JTS Topology Suite (also: Java Topology Suite)[2] [168]. It is published under the *GNU*

[2] `http://sourceforge.net/projects/jts-topo-suite/`

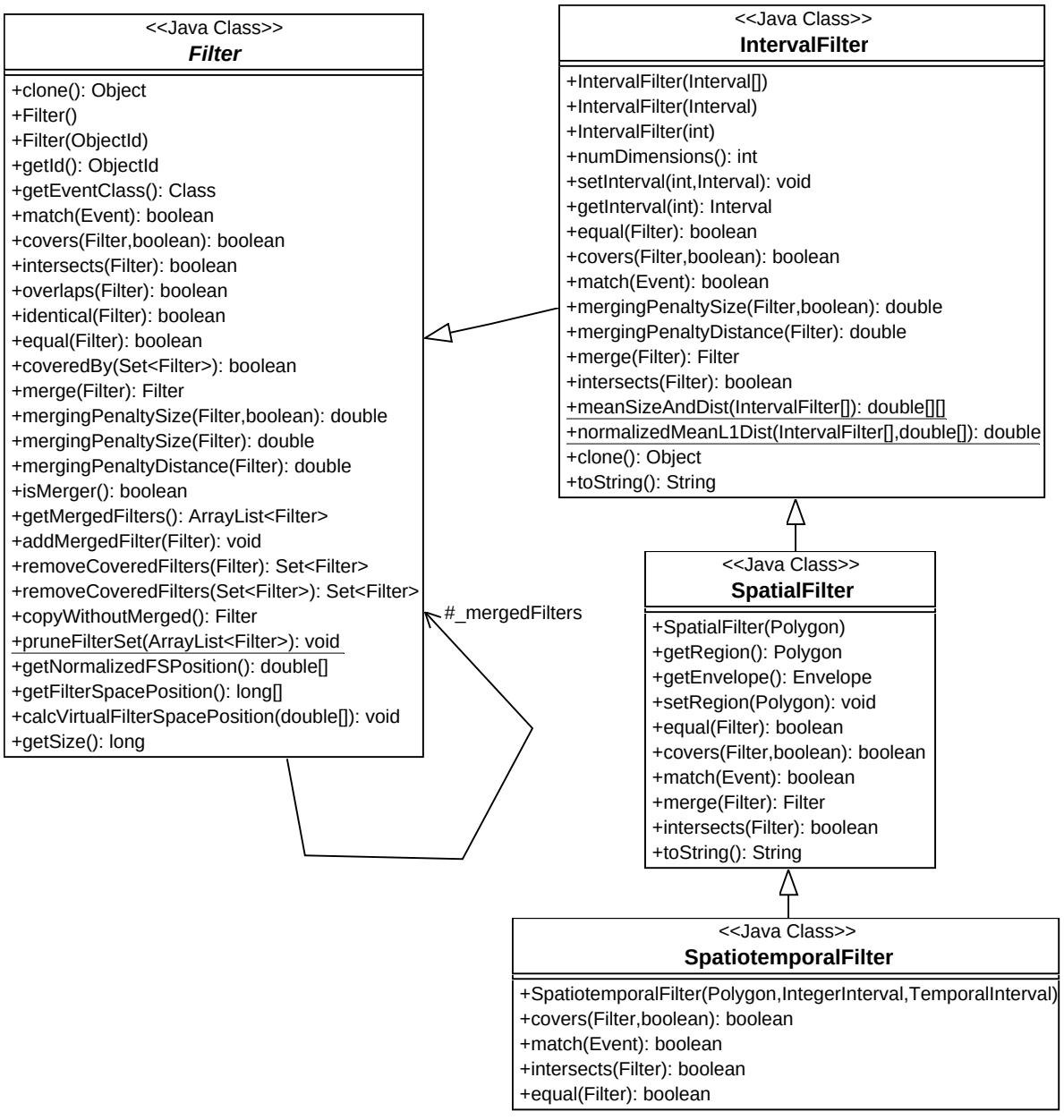


Figure A.5.: Classes `IntervalFilter`, `SpatialFilter`, and `SpatiotemporalFilter`

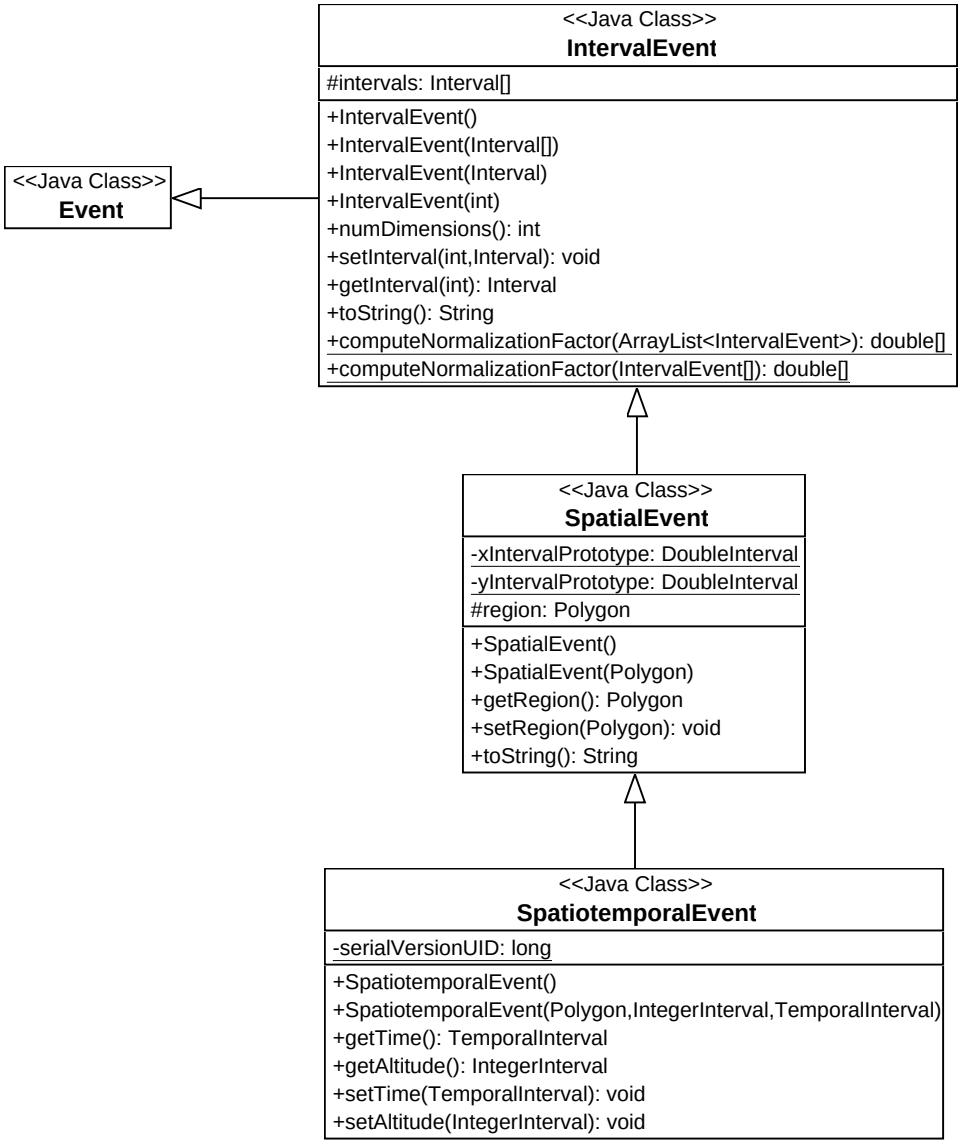


Figure A.6.: Classes `IntervalEvent`, `SpatialEvent`, and `SpatiotemporalEvent`

Lesser General Public License. JTS provides an object model for Euclidean planar linear geometry together with a set of fundamental geometric functions, implementing the spatial objects defined in the OGC Simple Feature Specification (SFS) [142], on which the aeronautical spatial model discussed in Section 3.3 is also based.

JTS uses the *Dimensionally Extended 9-Intersection Matrix (DE-9IM)* to describe and represent topolgical relationships of spatial objects according to the SFS. The DE-9IM is an extension of the 4-IM that we used to implement the 4-intersection logic. The relationships and matrix values for simple regions can be mapped smoothly between the two systems.

A.2.2 Intervals

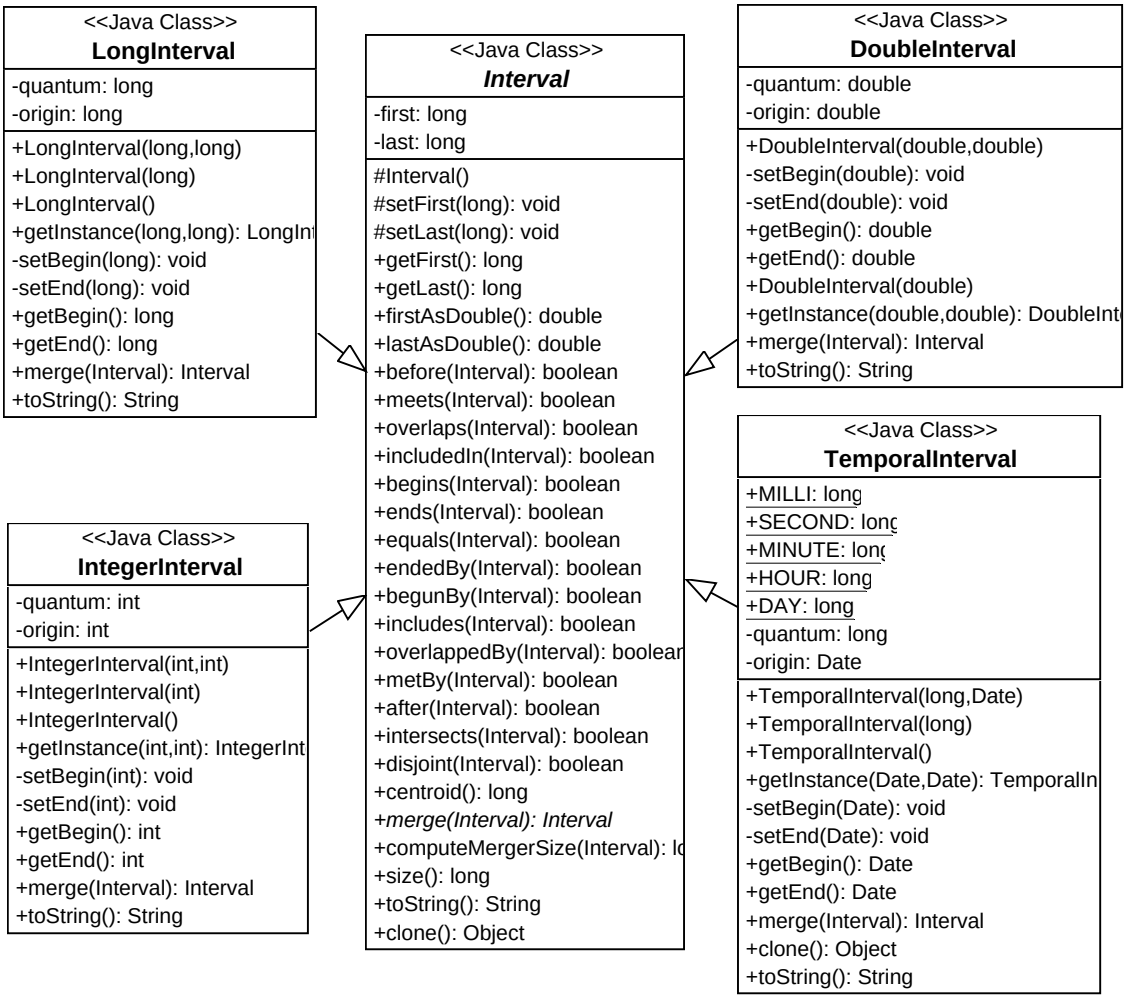


Figure A.7.: Interval Classes

The abstract base class `Interval` implements intervals over discrete point types (Figure A.7). Points are represented by `long` integer numbers, thus providing for 2^{64} discrete point values. The interval itself is represented by the two points `first` and `last`. Access to these points via the `getFirst()` and `getLast()` functions is limited to the concrete inheriting classes `IntegerInterval`, `LongInterval`, `DoubleInterval`, and `TemporalInterval`, each of which represents an own type of interval through virtual `begin` and `end` points in the respective type.

The framework implements the logic of discrete point types by mapping a value in the original space to a value in the `long` range. The mapping is defined by a `quantum` and an `origin` in the space of the original value. For instance, if the interval $[.3,.8]$ in $\mathbb{R}$ is mapped to an interval $\left[i^-,i^+\right)$ over discrete point types (represented as values in $\mathbb{N}$) using a discretization of the space in steps of 0.5 and an origin at -100.0, the interval is first translated such that the designated origin is at the origin of $\mathbb{N}$, 0, and the number of steps of length 0.5 are determined from the origin up to and including the begin and end values. Hence, $i^- = \lfloor(0.3-(-100))/0.5\rfloor$ and $i^+ = \lceil(0.8-(-100))/0.5\rceil$, such that the interval is represented as $\left[200,202\right)$. This is implemented in the `setBegin()` and `setEnd()` methods (Figure A.8).

```
private void setBegin ( double begin )
{
  super.setFirst ( ( long ) Math.floor ( ( begin - origin ) / quantum ) );
}
private void setEnd ( double end )
{
  super.setLast ( ( long ) Math.ceil ( ( end - origin ) / quantum ) );
}
```

Figure A.8.: Methods `setBegin` and `setEnd` of class `DoubleInterval` implementing the mapping between the original value space and the discrete point space

The definition of a particular interval thus requires on one hand to define the mapping to the discrete point space, which is specified by `quantum` and `origin`. On the other hand, the begin and end values in the original space then form the actual interval. Only intervals based on the same mapping can be compared sensibly. This is enforced in our implementation by providing the possibility to instantiate interval *prototypes* only. The constructor of each interval class takes the values of the quantum and the origin to define the mapping. Concrete instances of an interval based on this mapping can only be created using the `getInstance()` method supplying the `begin` and `end` values of the interval (Figure A.9).

```
public DoubleInterval ( double quantum, double origin )
{
  this.quantum = quantum;
  this.origin = origin;
}

public DoubleInterval getInstance ( double begin, double end )
{
  DoubleInterval di = ( DoubleInterval ) this.clone ();
  di.setBegin ( begin );
  di.setEnd ( end );
  return di;
}
```

Figure A.9.: Constructor and method `getInstance` of class `DoubleInterval` implementing the prototype scheme of intervals

With temporal intervals, our implementation applies the common Unix time: Timestamps are represented as milliseconds since January 1, 1970, 00:00h. The Java utilities class `Date` represents those timestamps. `TemporalInterval` therefore uses `Date` objects for the external representation of the begin and end points of intervals. Internally, it works with `long` values representing the timestamps. Chronons are specified in multiples of milliseconds, and publicly accessible constants for the common time units second, minute, hour,... are provided such that a mapping can for instance be defined by instantizing `new TemporalInterval(TemporalInterval.SECOND, new Date())` to define a mapping to a discrete time space with one second chronons and the origin at the moment of the instantization (`new Date()` returns a date object representing the current system time).

Interval Relationships

The base class `Interval` implements the interval algebra based on Allen's operators by providing boolean-valued functions for all possible relationships by their names. Figure A.10 shows the implementation of a selection of these relations and of the convenience named relations intersects and disjoint. Note that the implementation of intersects is based on our definition given in Equation (4.1)[3], thus valid only for the right-open intervals that we generally assume.

[3] Section 4.1.1, page 72

```
public boolean before ( Interval that )
{
  return ( this.last < that.first );
}
public boolean meets ( Interval that )
{
  return ( this.last == that.first );
}
public boolean overlaps ( Interval that )
{
  return (
   this.first < that.first
    && that.first < this.last
    && this.last < that.last
  );
}
public boolean intersects ( Interval that )
{
  return ( this.first < that.last && that.first < this.last );
}
public boolean disjoint ( Interval that )
{
  return ( !this.intersects ( that ) );
}
```

Figure A.10.: Implementation of Allen's operators and convenience named relations as boolean-valued functions in class `Interval`

Merging Operation

All concrete interval classes must implement the method `merge()` defined as abstract method in the base class `Interval` (Figure A.11). It takes as input argument another interval, and implements the interval merging operation $I_{\text{merger}} = I_{\text{this}} \sqcup I_{\text{that}}$. The returned interval object is a new object, `this` and `that` are not changed.

```
public Interval merge ( Interval that )
{
  if ( ( ! ( ( TemporalInterval ) that ).origin.equals ( this.origin ) )
   || ( ( ( TemporalInterval ) that ).quantum != this.quantum ) )
     return null;

  TemporalInterval merger = this.clone ();
  if ( this.getFirst () < that.getFirst () )
   merger.setFirst ( this.getFirst () );
  else
   merger.setFirst ( that.getFirst () );

  if ( this.getLast () > that.getLast () )
   merger.setLast ( this.getLast () );
  else
   merger.setLast ( that.getLast () );

  return merger;
}
```

Figure A.11.: Constructor and method `getInstance` of class `DoubleInterval` implementing the prototype scheme of intervals

The method first checks wether both intervals are based on the same mapping, before the interval merger is created by cloning `this` and setting the `begin` and `end` points according to Equation (5.1)[4].

[4] Section 5.1, page 97

B Merging Experiments

The merging strategy alternatives —for the merging penalty score function and the merging candidate search— were tested in the experiments described in Section 5.4 using sets of randomly generated 1-intervals and 2-intervals (rectangles) with varying characteristics.

The sets of filters $\mathscr{G}$ and notifications $\mathscr{N}$ that are used in the experiments were created from these sets of (2-)intervals. In every run of the experiment, the set $\mathscr{G}$ consisted of 10,000 filters (except for the experiments to determine the effect of input filter set size), and the set $\mathscr{N}$ consisted of 100,000 notifications, and the experimental algorithm (Figure 5.13) was executed for a large number of different merging penalty score threshold values.

In the following, the composition of the synthetic sample data sets and all significant results from the experiments are presented, a selection of which has been used in Section 5.4 to discuss the different characteristics of the approaches and impact of the data sets.

The results are in the following consistently presented as plots of precision over reduction. The plots for the size-based penalty score functions are truncated where the reduction boosts to 1 for the reasons described in Section 5.4.3. The plots for the distance-based penalty score functions are truncated at reduction $r = 0.9$. At this point the runs of the experiment algorithm were stopped for time reasons, because the reduction approaches 1, and the precision approaches 0, asymptotically. Note that the plot axes vary. For the size-based penalty score functions, only a section of the reduction range $0 < r < 1$ and precision range $0 < p < 1$ is displayed.

B.1 1-Interval Sets

Eleven sets $\mathscr{I}_1, \ldots, \mathscr{I}_{11}$ of integer intervals $I = \left[i^-, i^+\right)$ in the value range $\left[0, 2^{20}\right]$ were generated. The experiments were carried out on these sets using as alternatives for the merging penalty score function *MP*:

- size penalty score s
- distance penalty score d,

and for the merging candidate search function *findMergingCandidate*, the exhaustive and the non-exhaustive alternative.

B.1.1 Sample Set Composition

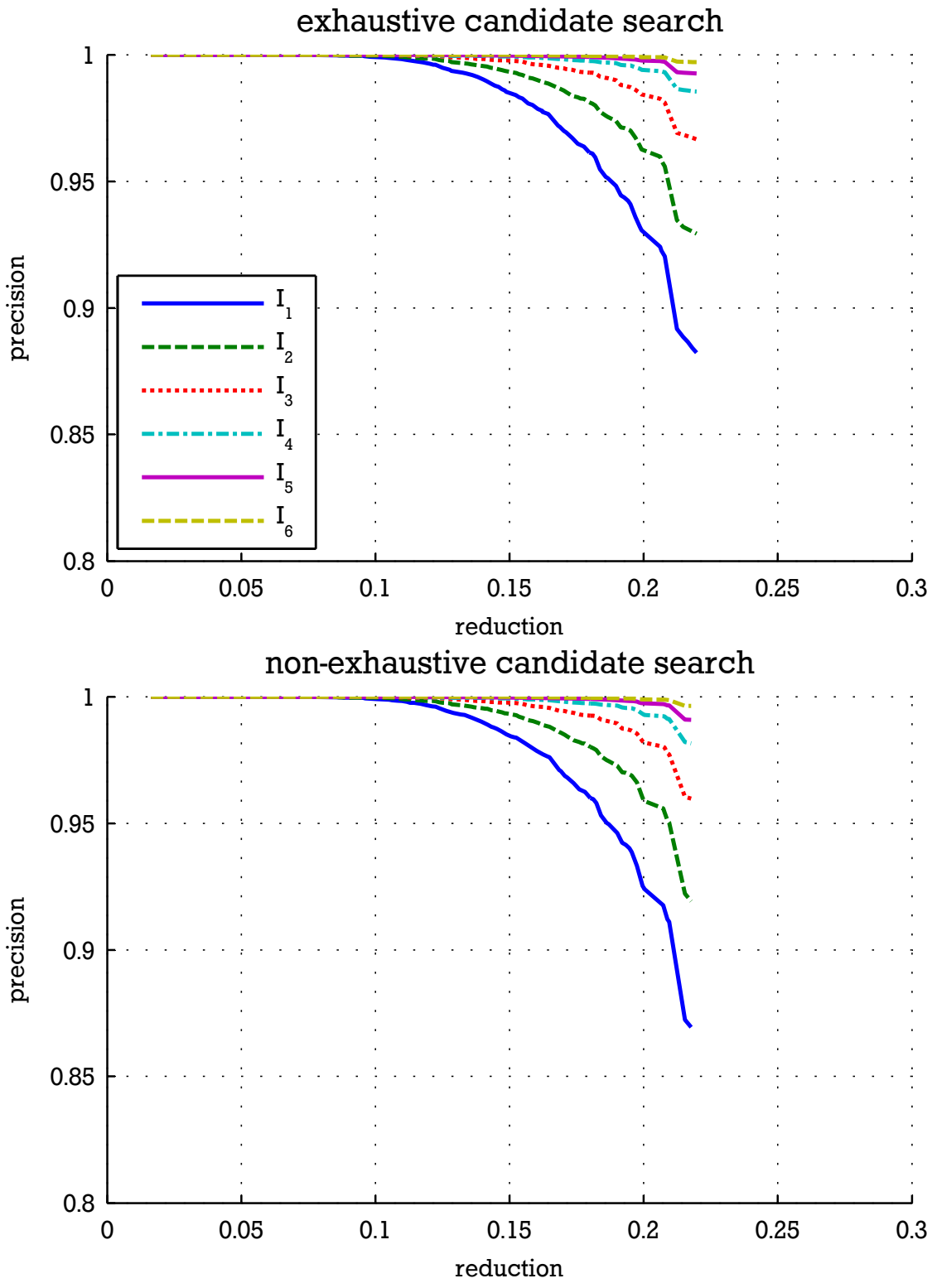
In sets $\mathscr{I}_1$ through $\mathscr{I}_6$, the intervals are uniformly distributed in the value range. The interval sizes $|I|$ follow a half-normal distribution (only the positive values in a normal distribution with $\mu = 0$) shifted by $+1$ to have only sizes ≥ 1) with different standard deviations $\sigma_{|I|}$. In set $\mathscr{I}_1$, the standard deviation of the interval length is $\sigma_{|I|} = 2^3$, i.e., 68 % of all intervals are smaller than 9. In set $\mathscr{I}_2$ the standard deviation is $\sigma_{|I|} = 2^4$, in set $\mathscr{I}_3$ it is $\sigma_{|I|} = 2^5$ and so on.

In sets $\mathscr{I}_7$ through $\mathscr{I}_{12}$, the intervals are clustered around 10 hotspots, which are randomly (uniformly) distributed points $h_0, \ldots, h_9$ in the value range and are the same over all sets. The hotspot concentration is thus that the medians of the intervals are normally distributed with $\mu = h_i$ and $\sigma = 2^{15}$. Hence, approximately 68 % of all intervals are closer than 32,768 to a hotspot. The sets differ in the interval size, which is distributed as in the sets $\mathscr{I}_1$ through $\mathscr{I}_6$ with a standard deviation of $\sigma_{|I|} = 2^3$ in $\mathscr{I}_7$, and so on until $\sigma_{|I|} = 2^7$ in $\mathscr{I}_{11}$.

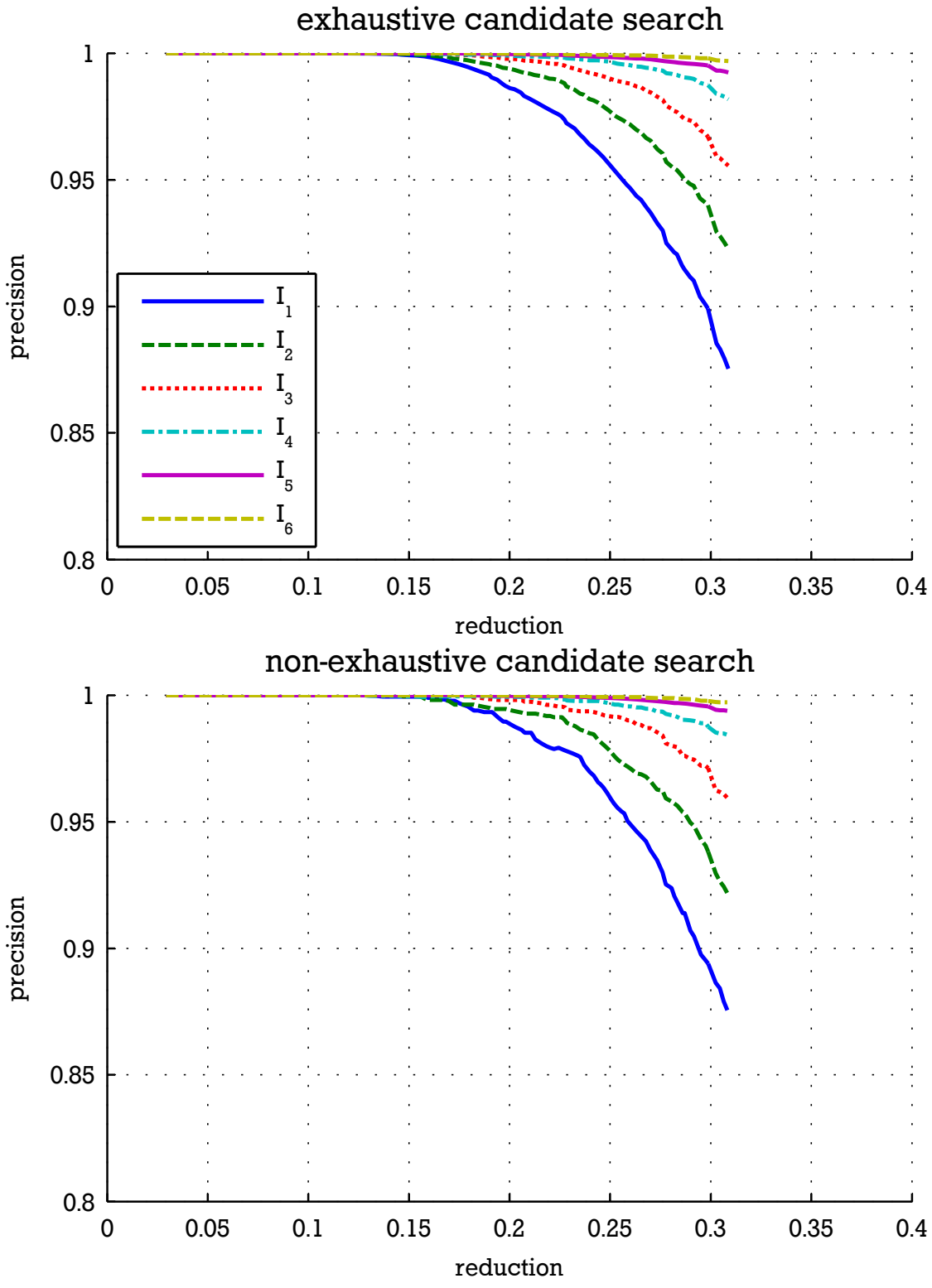
Table B.1.: Synthetic sample sets of 1-intervals

size	uniform distribution	clustered distribution
$\sigma_{\|I\|} = 8$	$\mathscr{I}_1$	$\mathscr{I}_7$
$\sigma_{\|I\|} = 16$	$\mathscr{I}_2$	$\mathscr{I}_8$
$\sigma_{\|I\|} = 32$	$\mathscr{I}_3$	$\mathscr{I}_9$
$\sigma_{\|I\|} = 64$	$\mathscr{I}_4$	$\mathscr{I}_{10}$
$\sigma_{\|I\|} = 128$	$\mathscr{I}_5$	$\mathscr{I}_{11}$
$\sigma_{\|I\|} = 256$	$\mathscr{I}_6$	$\mathscr{I}_{12}$

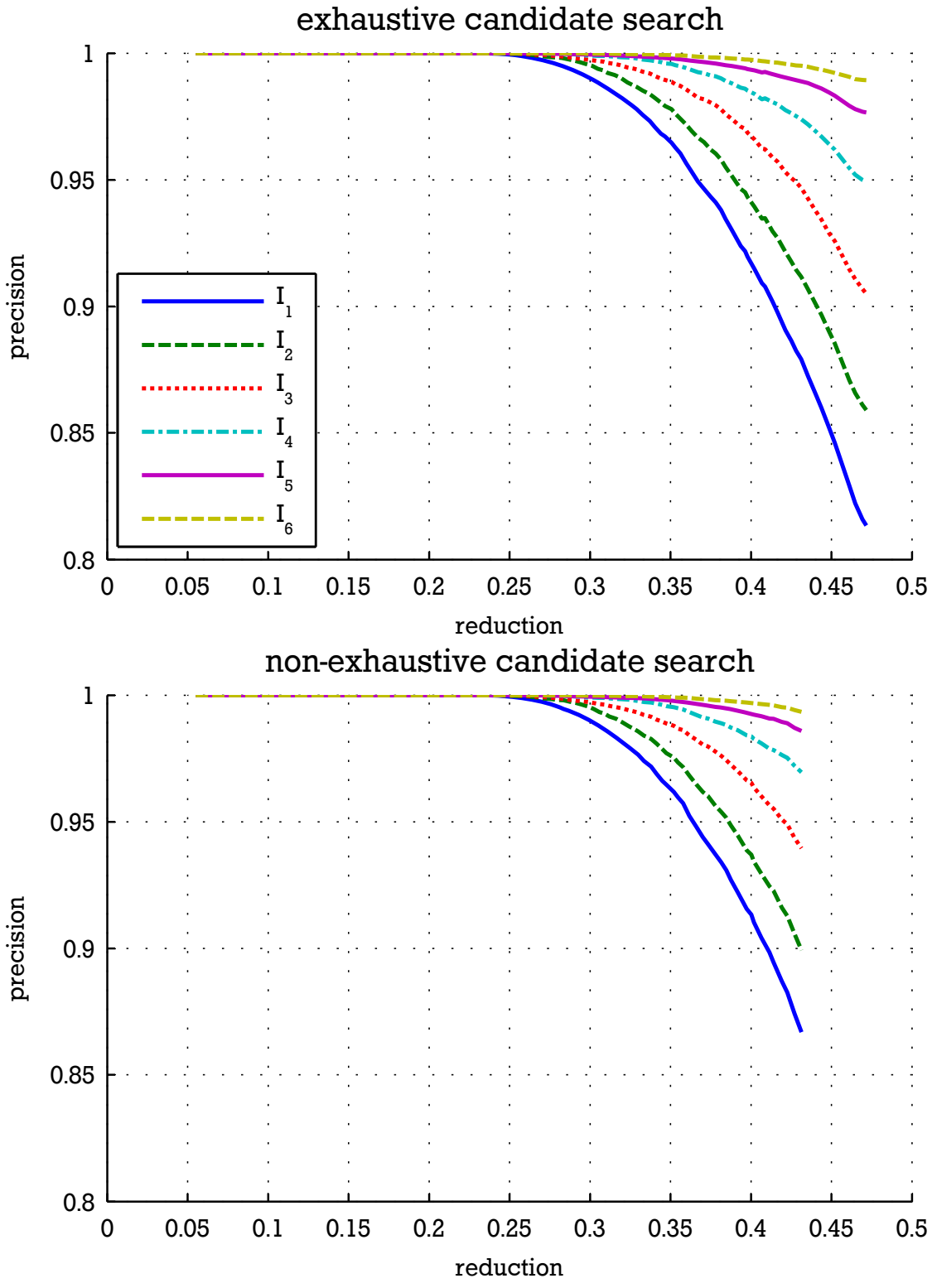
B.1.2 Size Penalty Score



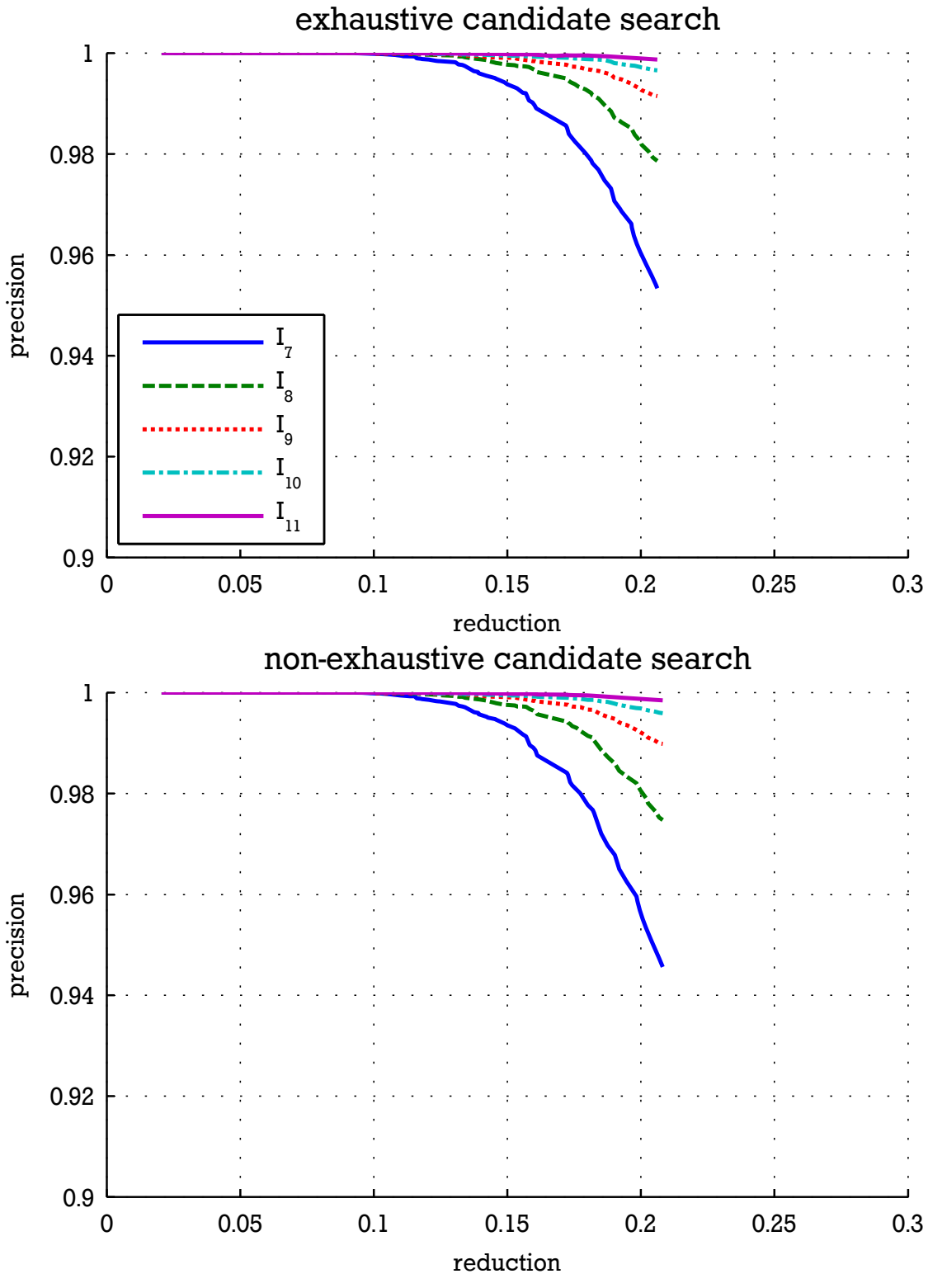
Size penalty score with filters from interval set $\mathscr{I}_1$ and notifications from interval sets as labeled in plot legend



Size penalty score with filters from interval set $\mathscr{I}_2$ and notifications from interval sets as labeled in plot legend

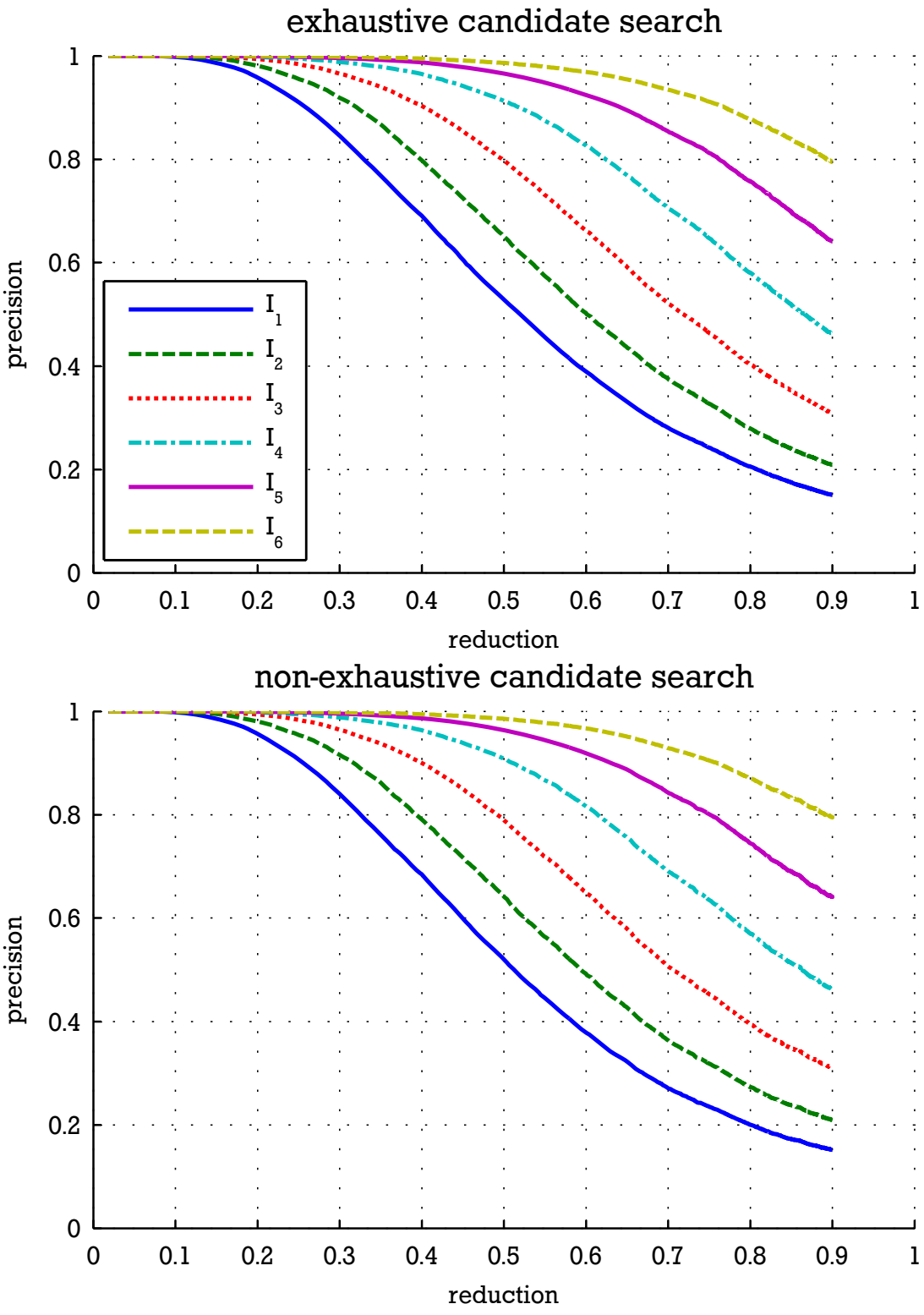


Size penalty score with filters from interval set $\mathscr{I}_3$ and notifications from interval sets as labeled in plot legend

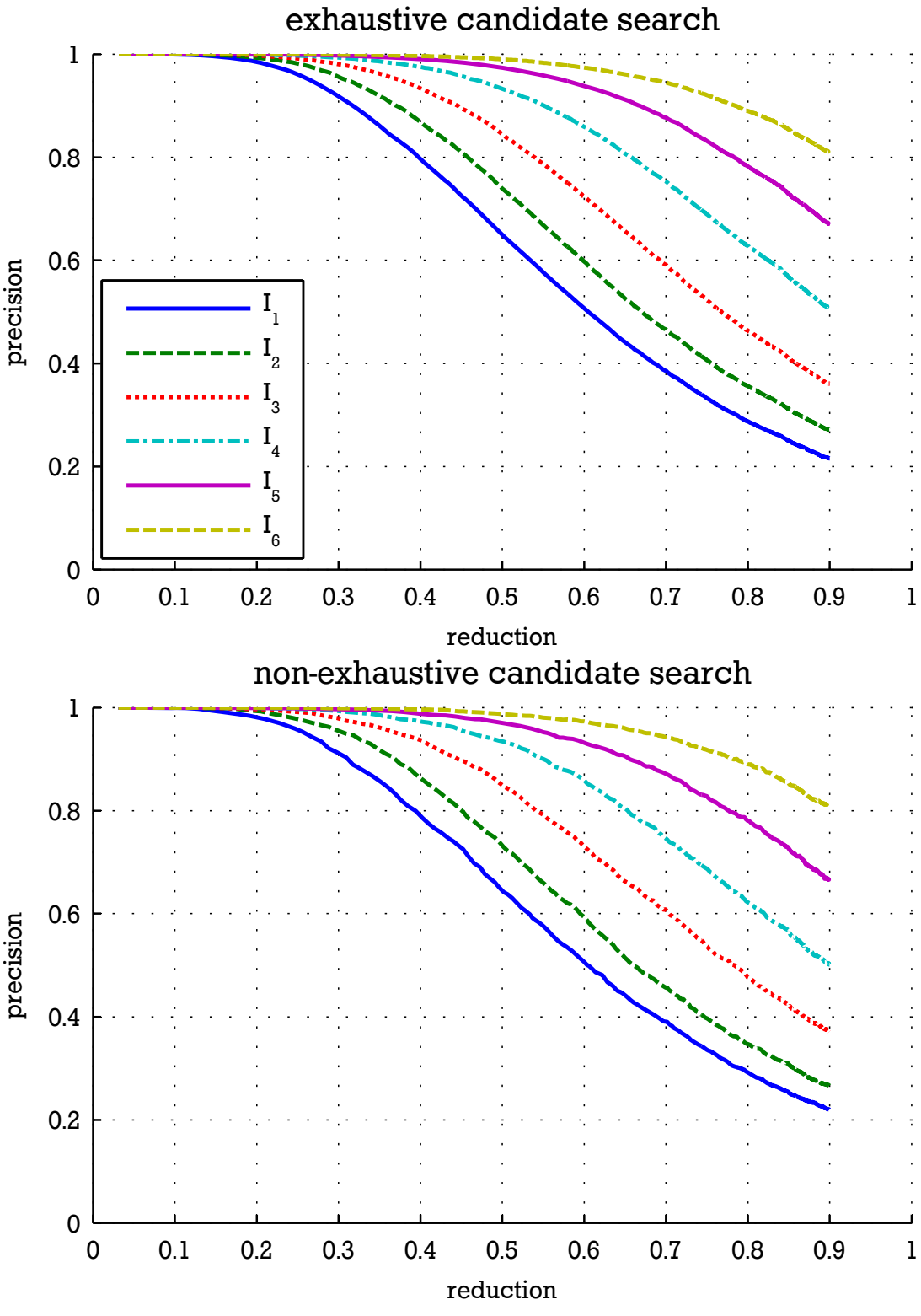


Size penalty score with filters from interval set $\mathscr{I}_7$ and notifications from interval sets as labeled in plot legend

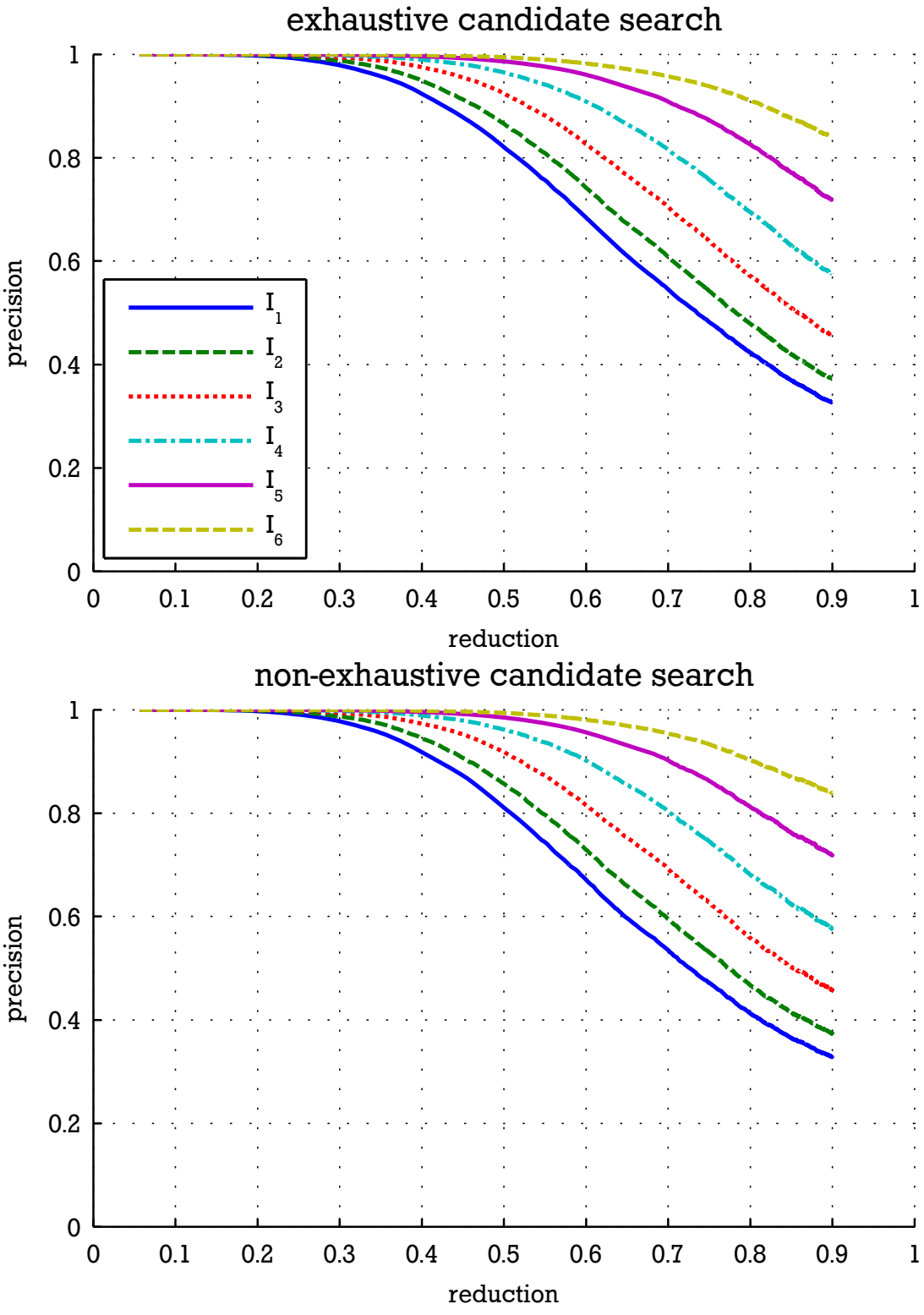
B.1.3 Distance Penalty Score



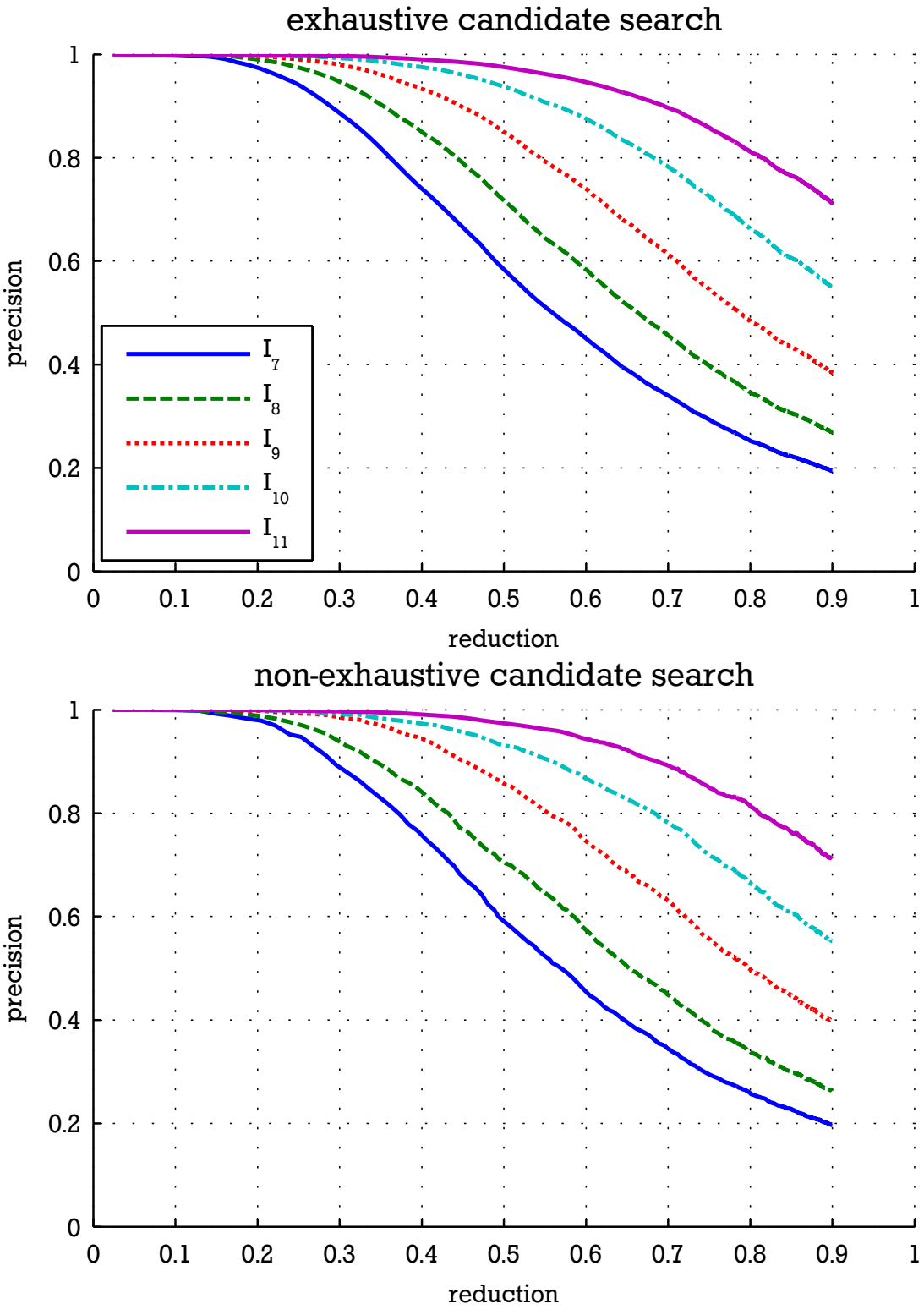
Distance penalty score with filters from interval set $\mathscr{I}_1$ and notifications from interval sets as labeled in plot legend



Distance penalty score with filters from interval set $\mathscr{I}_2$ and notifications from interval sets as labeled in plot legend



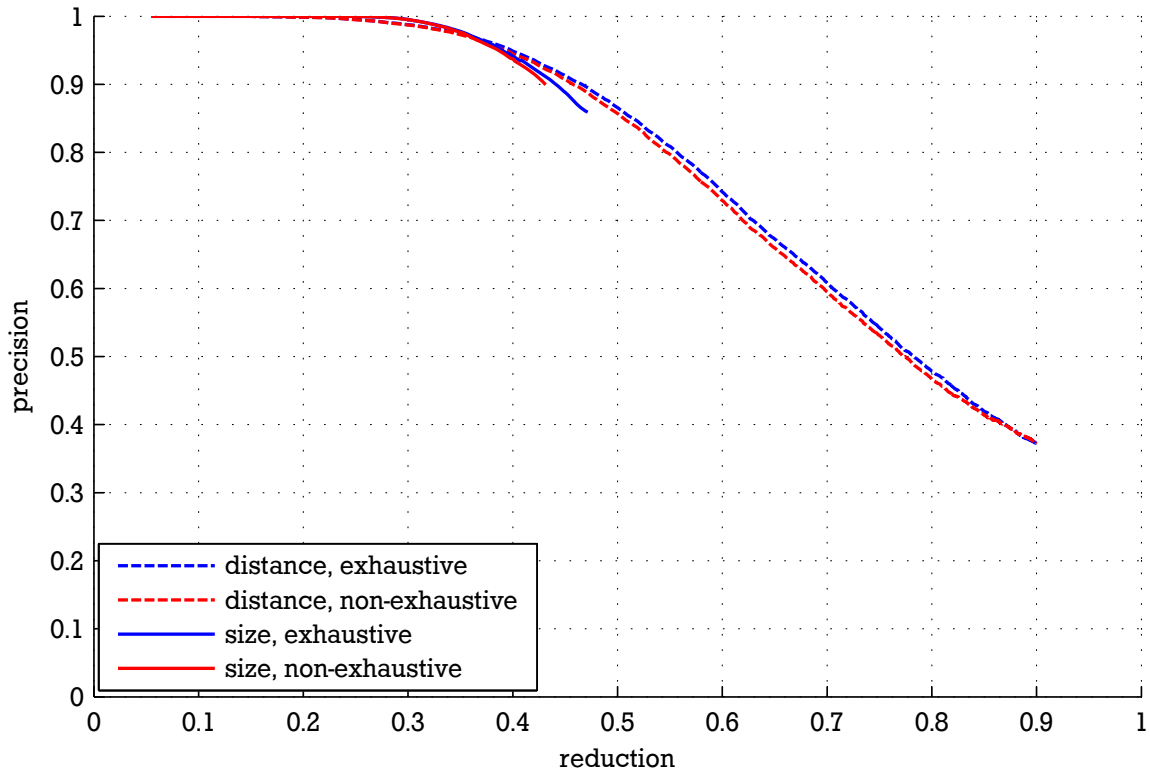
Distance penalty score with filters from interval set $\mathscr{I}_3$ and notifications from interval sets as labeled in plot legend



Distance penalty score with filters from interval set $\mathscr{I}_7$ and notifications from interval sets as labeled in plot legend

B.1.4 Direct Comparison of Strategies

The following figure shows plots for the different strategies. The filters were created from set $\mathscr{I}_3$, and the notifications from set $\mathscr{I}_1$. The sets were chosen such that the plots are typical for the strategies. The figure allows a direct comparison of the different merging strategies for 1-interval filters.



Exemplary direct comparison of size penalty *vs.* distance penalty and exhaustive *vs.* non-exhaustive candidate search approach with filters from interval set $\mathscr{I}_3$ and notifications from interval set $\mathscr{I}_1$

B.2 2-Interval Sets with Equal Dimension Characteristics

Six sets $\mathscr{R}_1, \ldots, \mathscr{R}_6$ of 2-dimensional integer intervals $R = \left(\left[r_1^-, r_1^+\right), \left[r_2^-, r_2^+\right)\right)$ were generated, and experiments were carried out on these sets using as alternatives for the merging penalty score function *MP*:

- size penalty score s
- mean size penalty score s_{mean}
- distance penalty score d

The non-exhaustive merging candidate search was used.

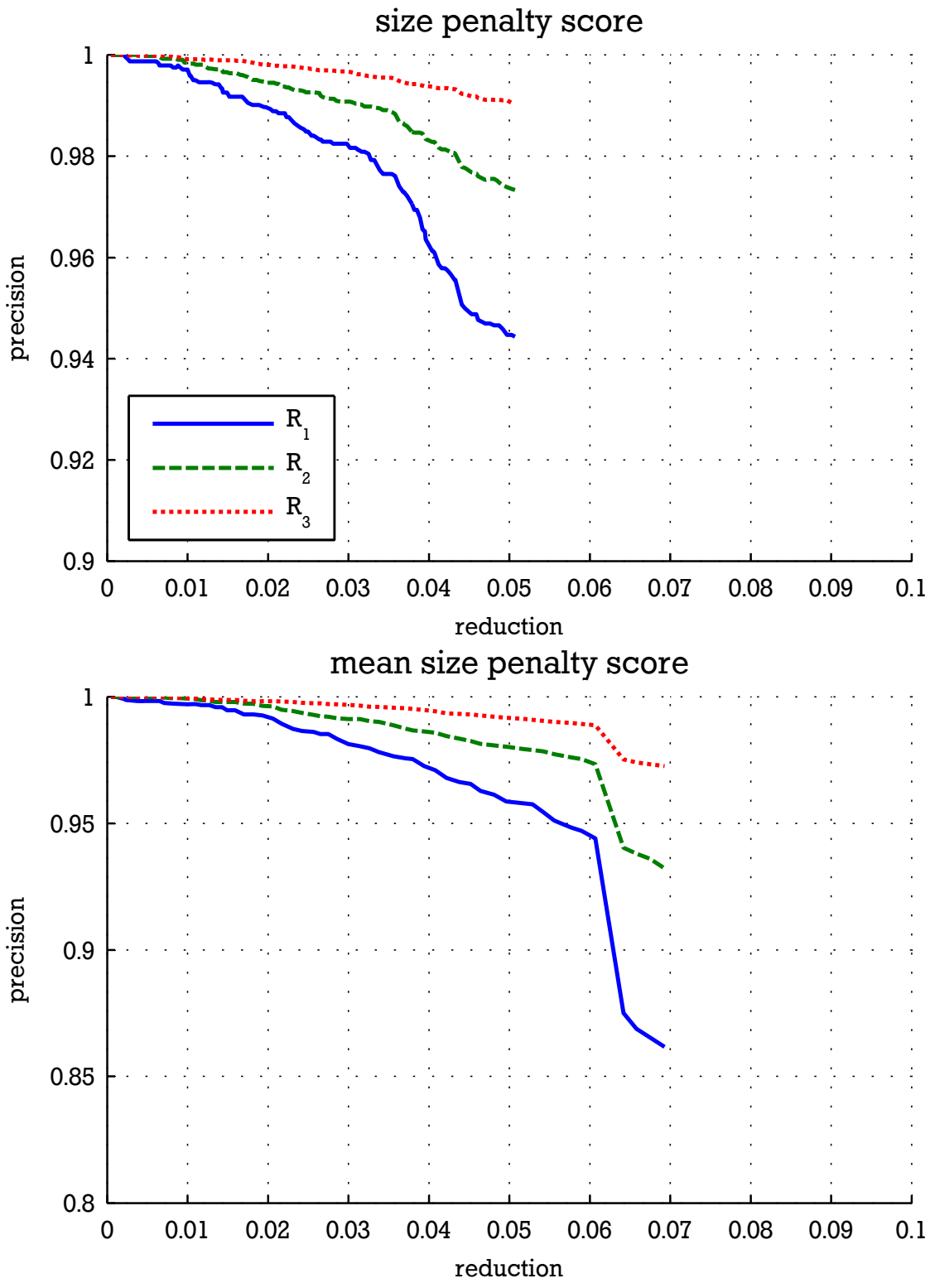
B.2.1 Sample Set Composition

In all sets $\mathscr{R}_1, \ldots, \mathscr{R}_6$, the intervals are in both dimensions in the value range $[0, 2^{20}]$, and the interval sizes $|I|$ follow a half-normal distribution, the standard deviation of which $\sigma_{|I|}$ is equal in both dimensions in each set. In sets $\mathscr{R}_1$ and $\mathscr{R}_4$, it is $\sigma_{|I|} = 2^{10}$; in $\mathscr{R}_2$ and $\mathscr{R}_5$, it is $\sigma_{|I|} = 2^{11}$; in $\mathscr{R}_3$ and $\mathscr{R}_6$, it is $\sigma_{|I|} = 2^{12}$. In sets $\mathscr{R}_1, \mathscr{R}_2, \mathscr{R}_3$, the intervals are uniformly distributed in the value range in both dimensions, whereas in sets $\mathscr{R}_4, \mathscr{R}_5, \mathscr{R}_6$, they are clustered in each dimension around 10 hotspots $h_0, \ldots, h_9$ in a normal distribution with $\mu = h_i$ and $\sigma = 2^{16}$.

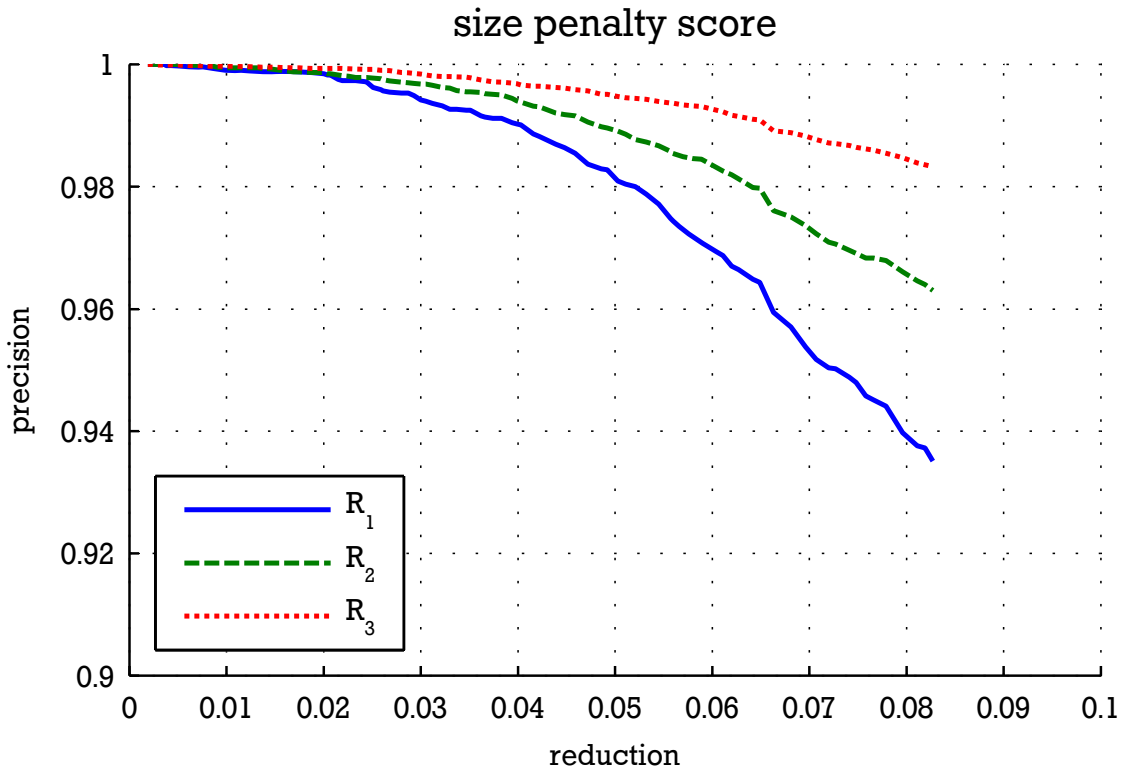
Table B.2.: Synthetic sample sets of 2-intervals with equal characteristics in both dimensions

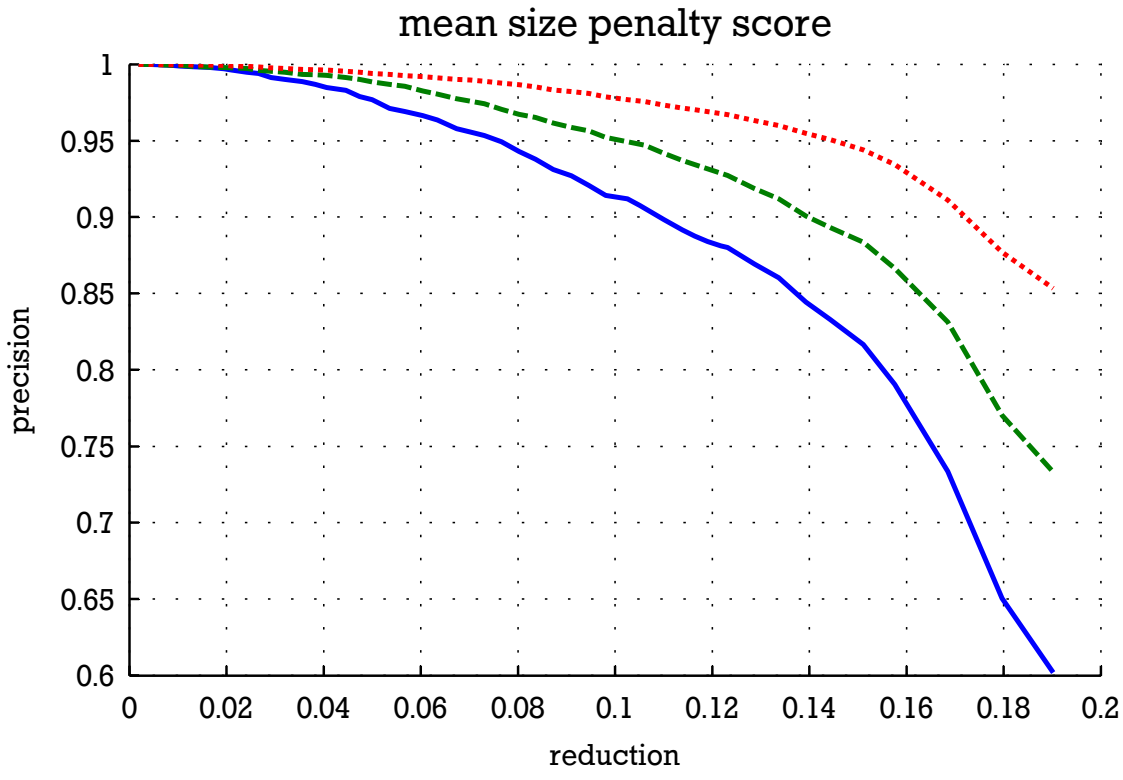
	$\sigma_{\|I\|} = 2^{10}$	$\sigma_{\|I\|} = 2^{11}$	$\sigma_{\|I\|} = 2^{12}$
uniform distribution	$\mathscr{R}_1$	$\mathscr{R}_2$	$\mathscr{R}_3$
clustered distribution	$\mathscr{R}_4$	$\mathscr{R}_5$	$\mathscr{R}_6$

B.2.2 Size-based Penalty Scores

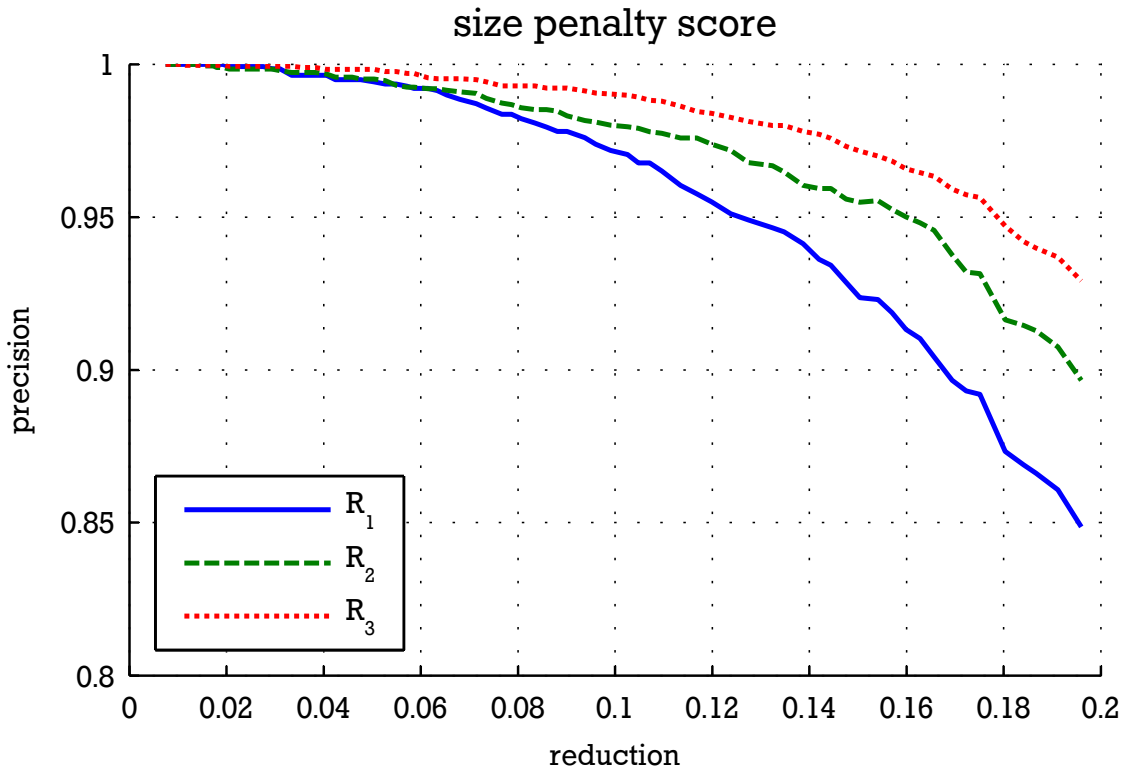


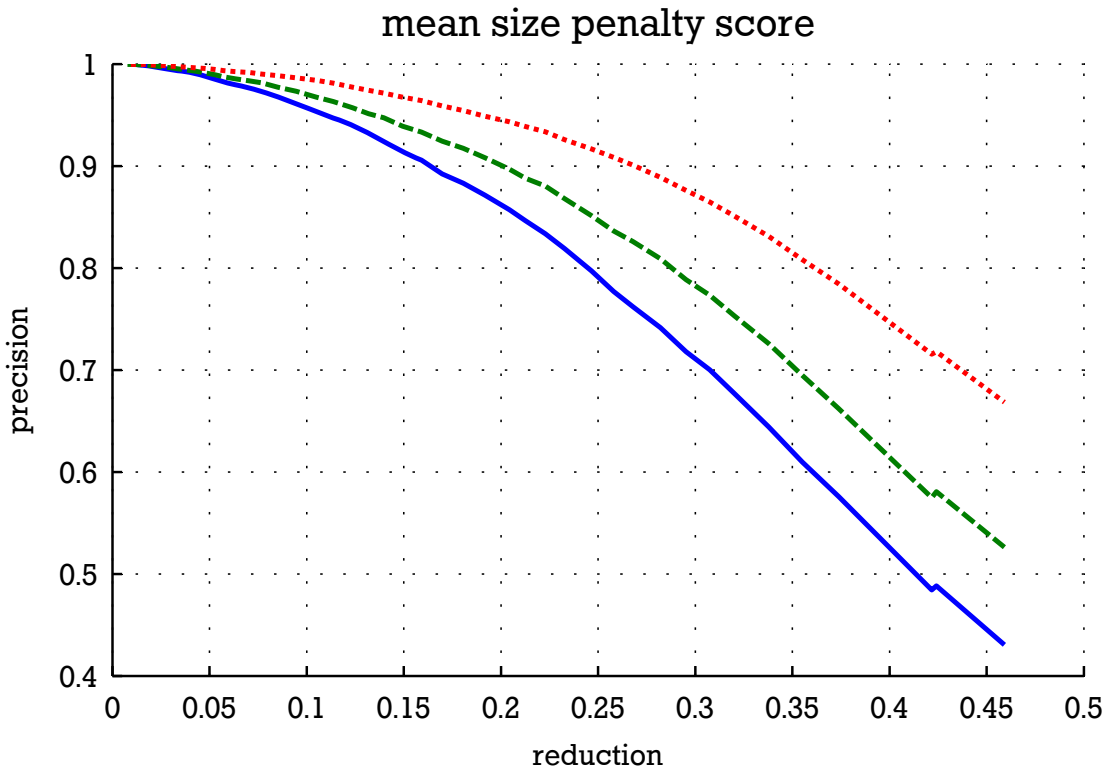
Size-based penalty scores with filters from 2-interval set $\mathscr{R}_1$ and notifications from sets as labeled in plot legend



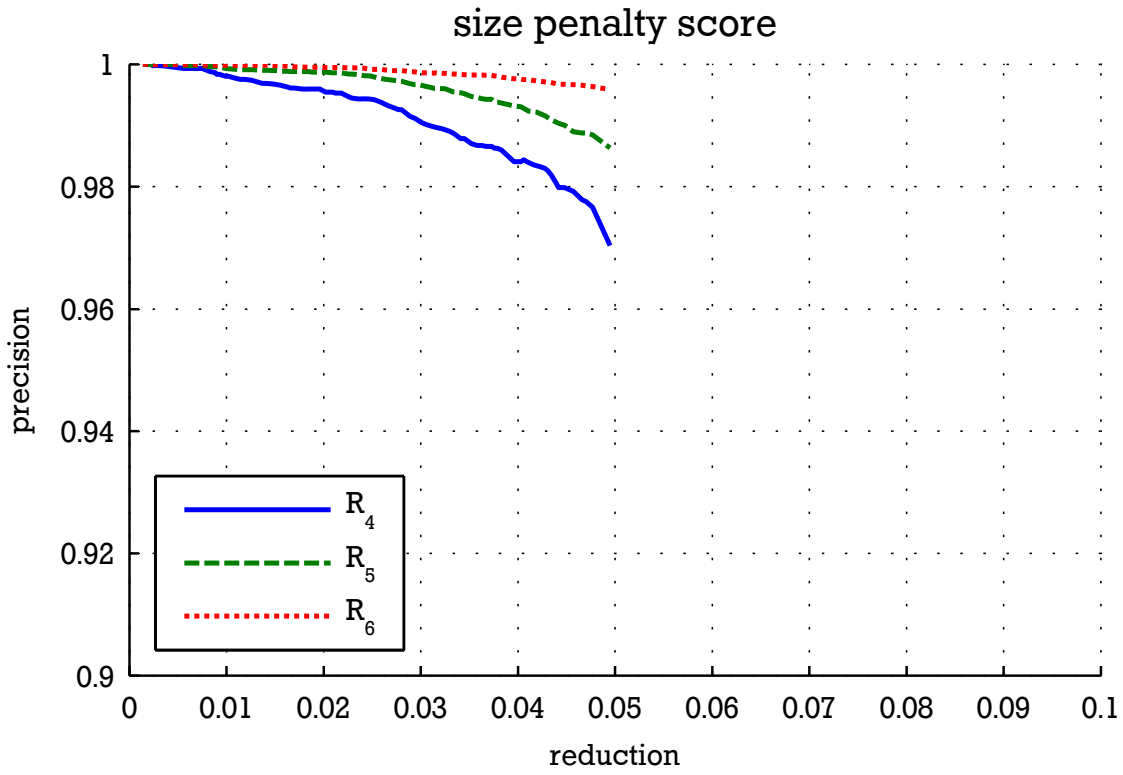


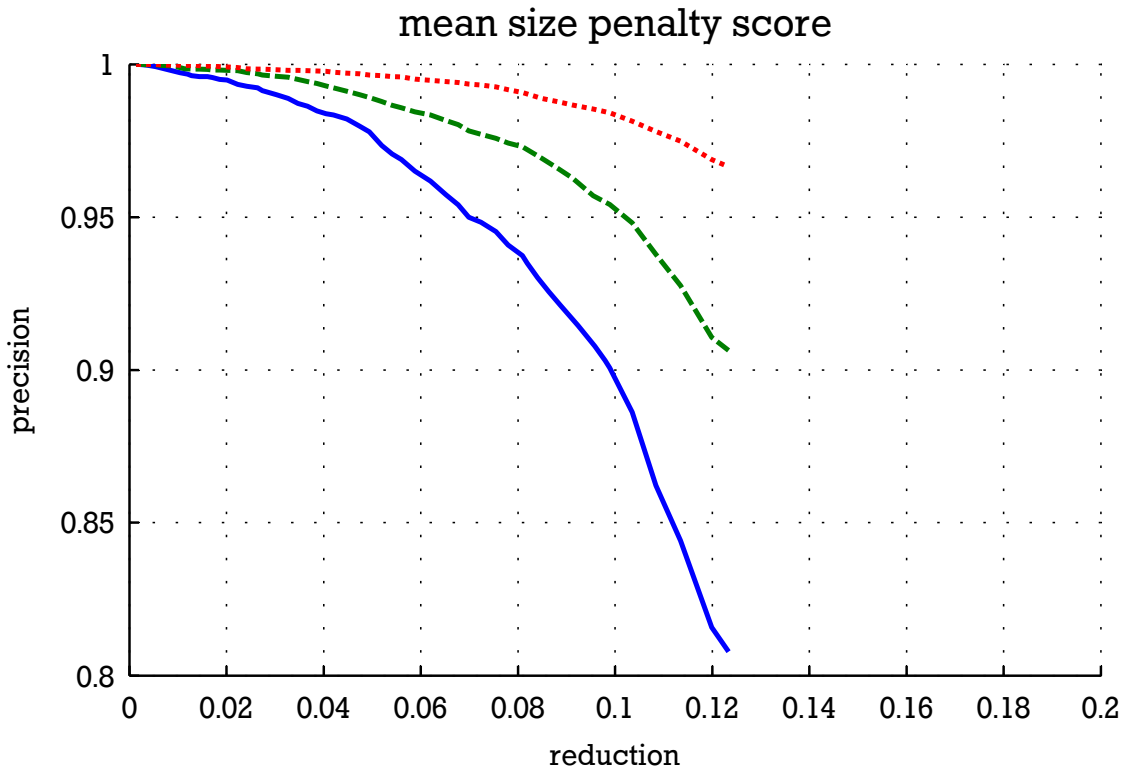
Size-based penalty scores with filters from 2-interval set $\mathscr{R}_2$ and notifications from sets as labeled in plot legend



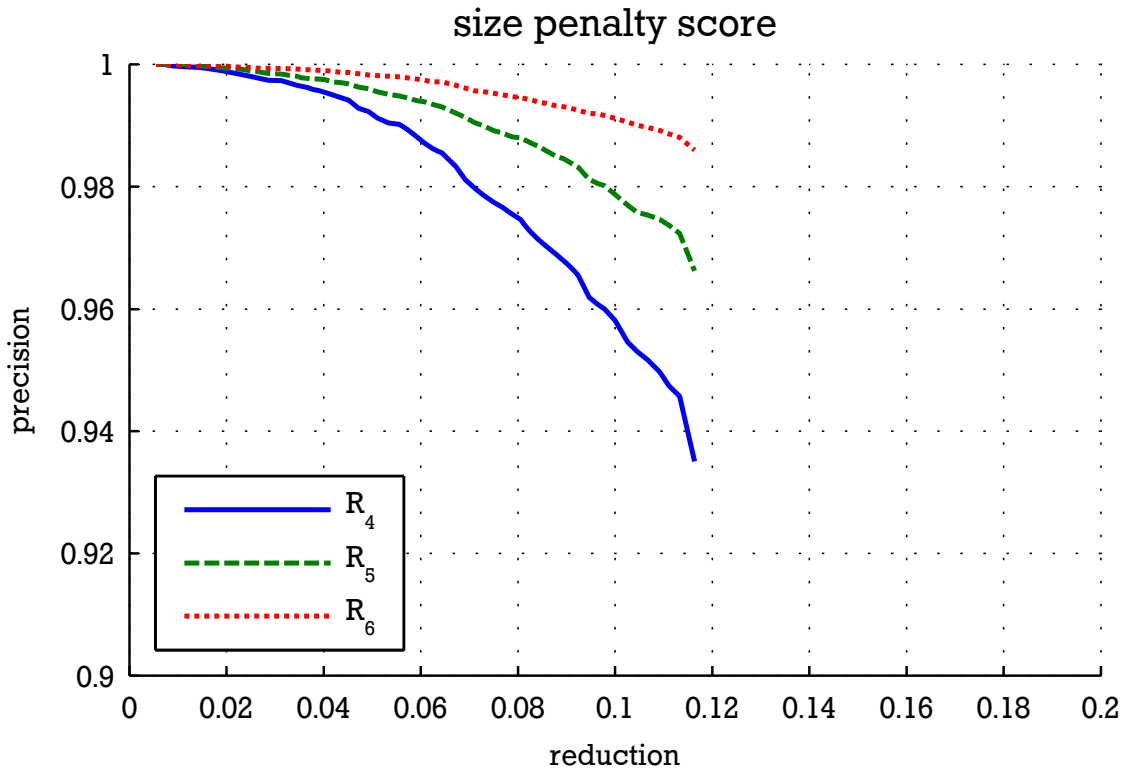


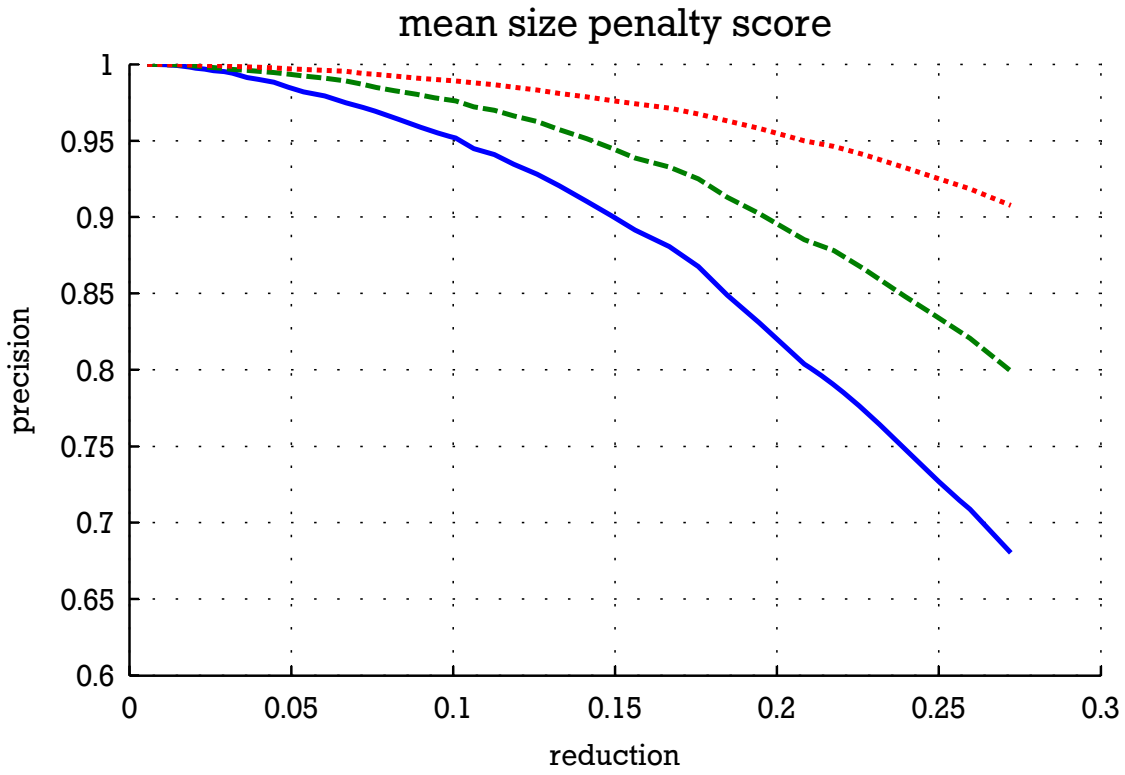
Size-based penalty scores with filters from 2-interval set $\mathscr{R}_3$ and notifications from sets as labeled in plot legend



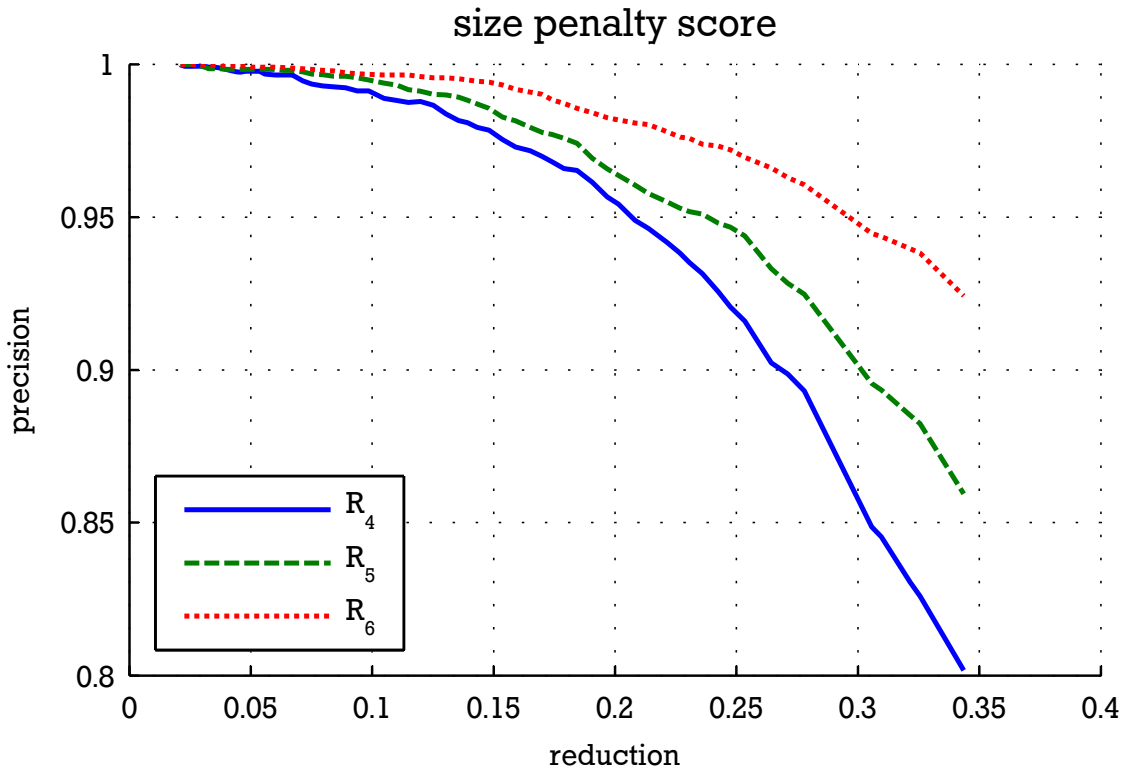


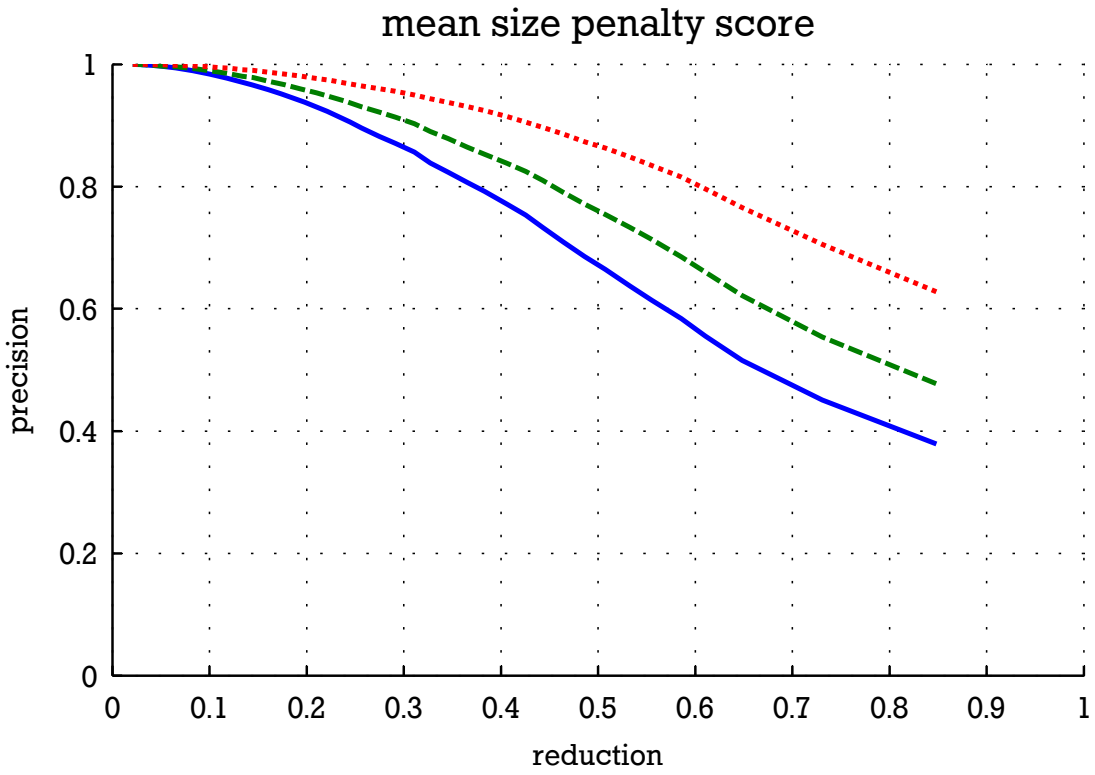
Size-based penalty scores with filters from 2-interval set $\mathscr{R}_4$ and notifications from sets as labeled in plot legend





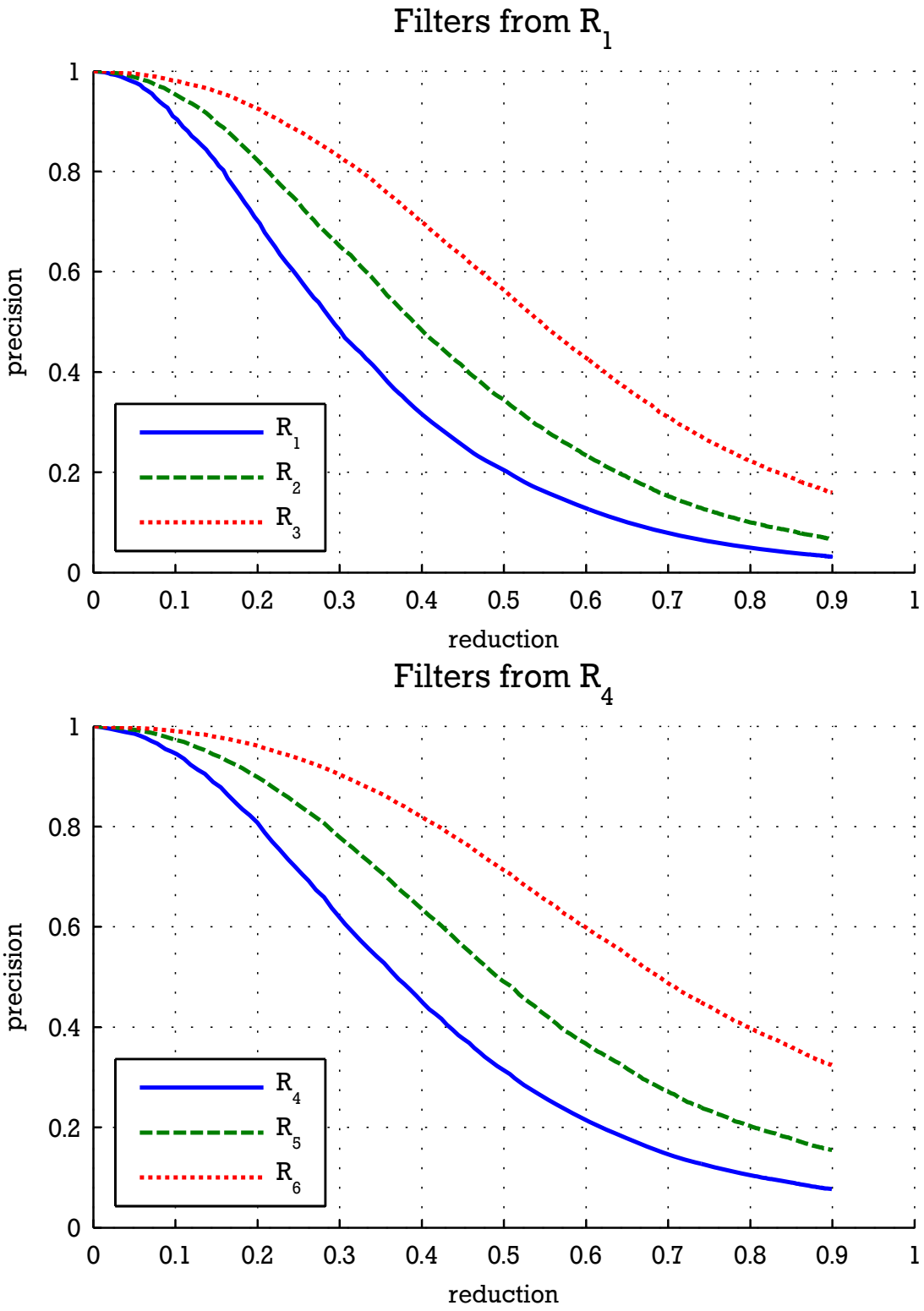
Size-based penalty scores with filters from 2-interval set $\mathscr{R}_5$ and notifications from sets as labeled in plot legend



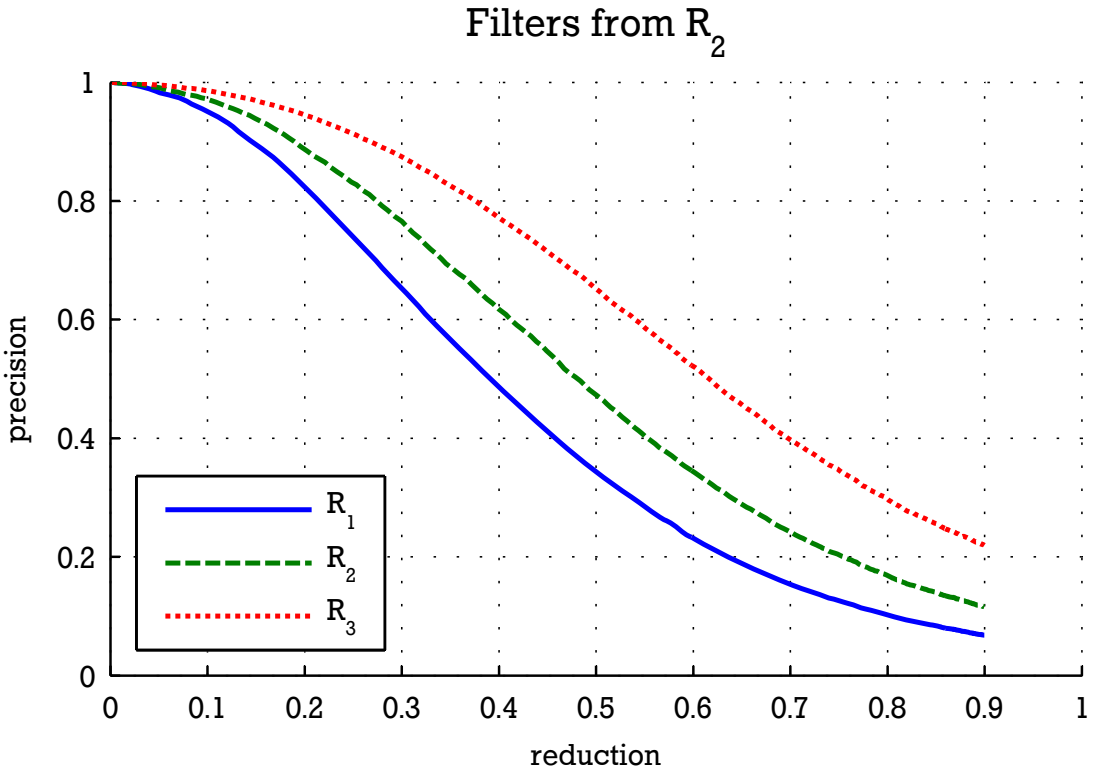


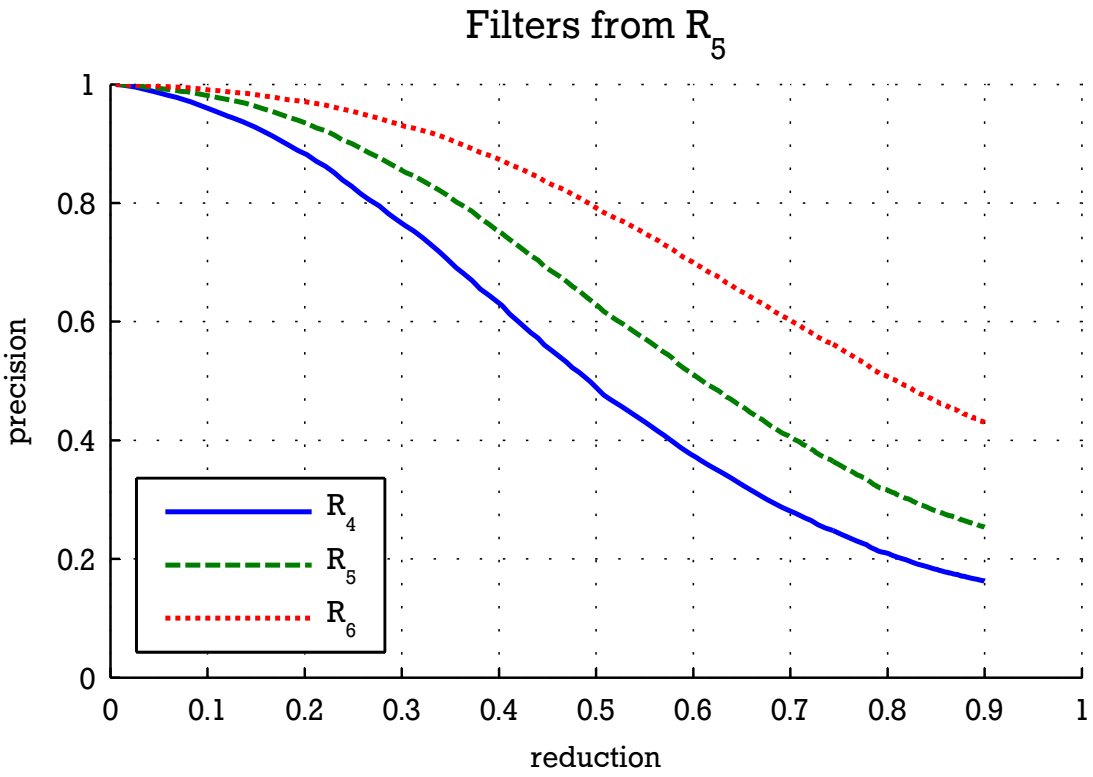
Size-based penalty scores with filters from 2-interval set $\mathscr{R}_6$ and notifications from sets as labeled in plot legend

B.2.3 Distance Penalty Score

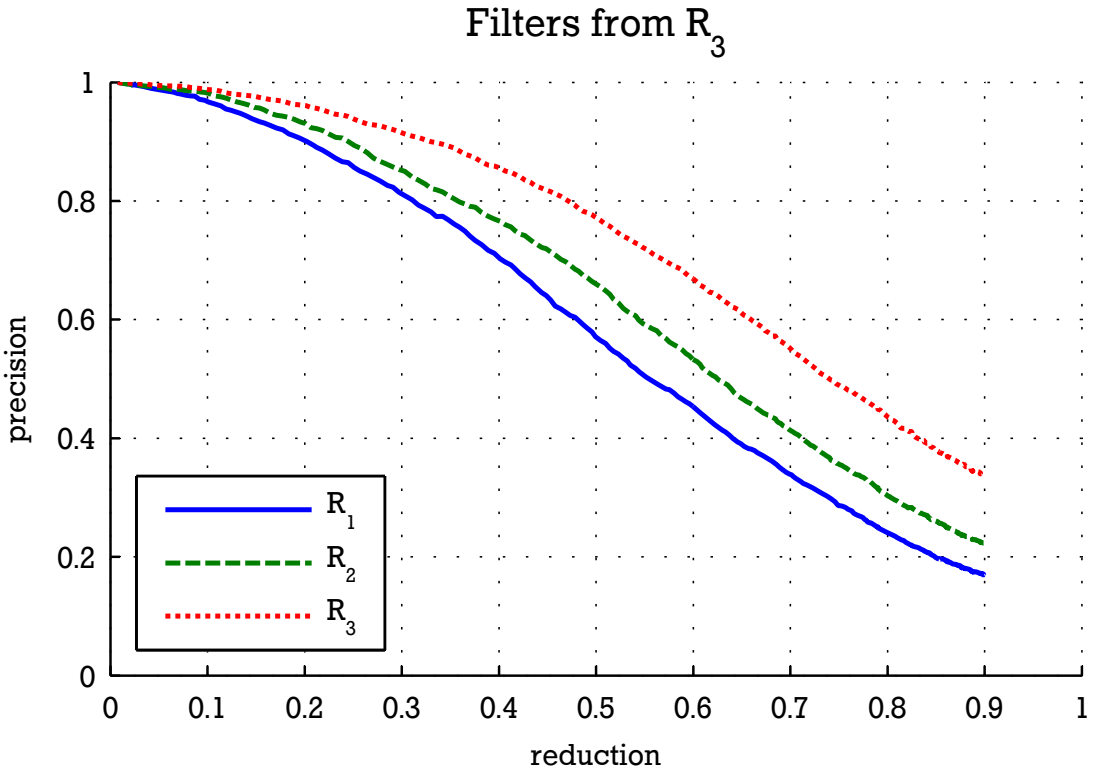


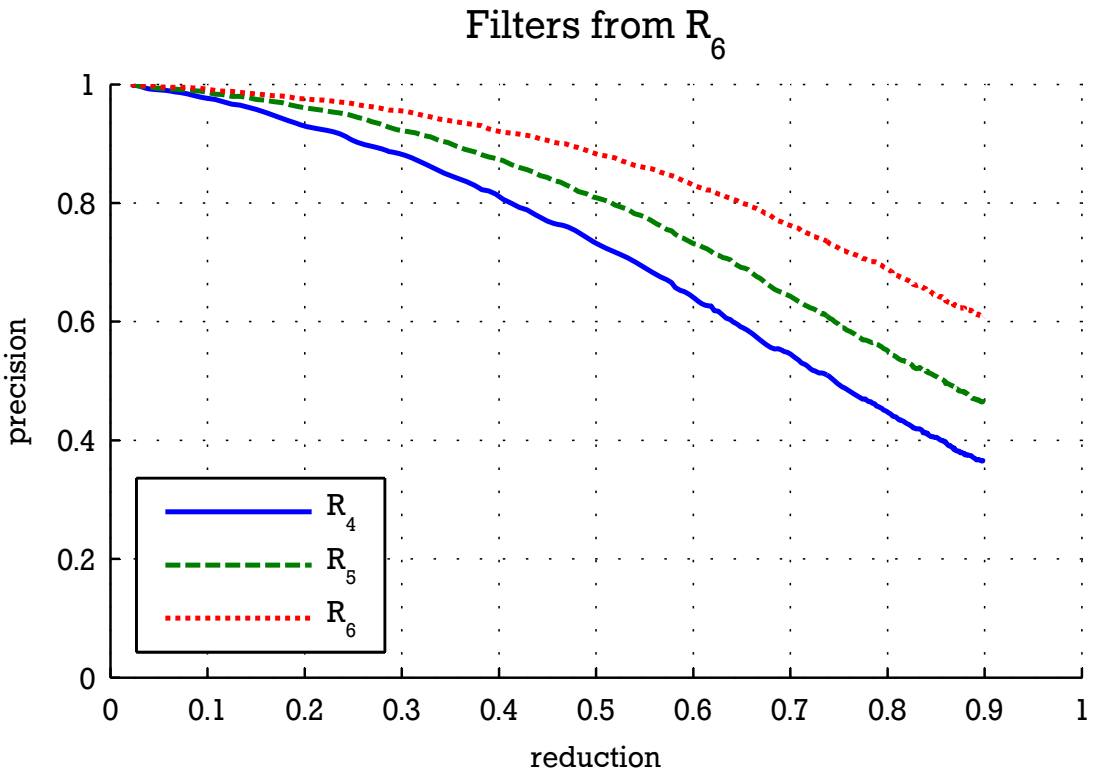
Distance penalty scores with filters from 2-interval sets $\mathscr{R}_1$ and $\mathscr{R}_4$ and notifications from sets as labeled in plot legend





Distance penalty scores with filters from 2-interval sets $\mathscr{R}_2$ and $\mathscr{R}_5$ and notifications from sets as labeled in plot legend

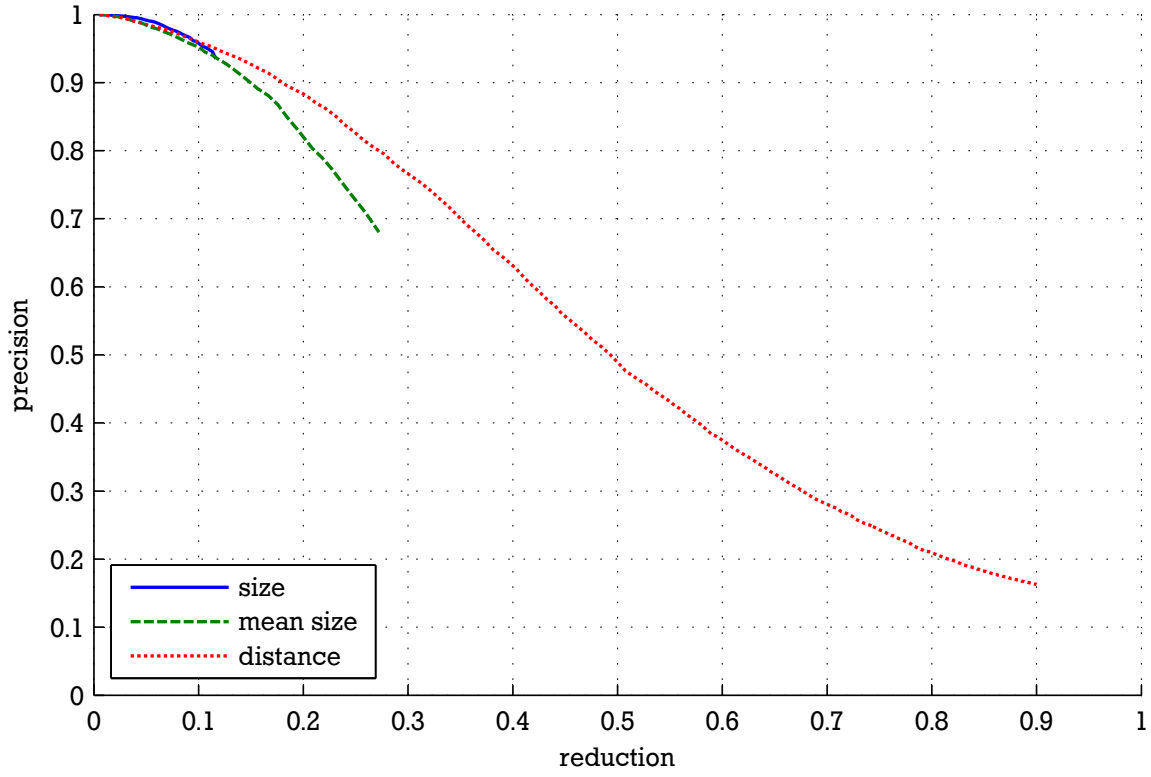




Distance penalty scores with filters from 2-interval sets $\mathscr{R}_3$ and $\mathscr{R}_6$ and notifications from sets as labeled in plot legend

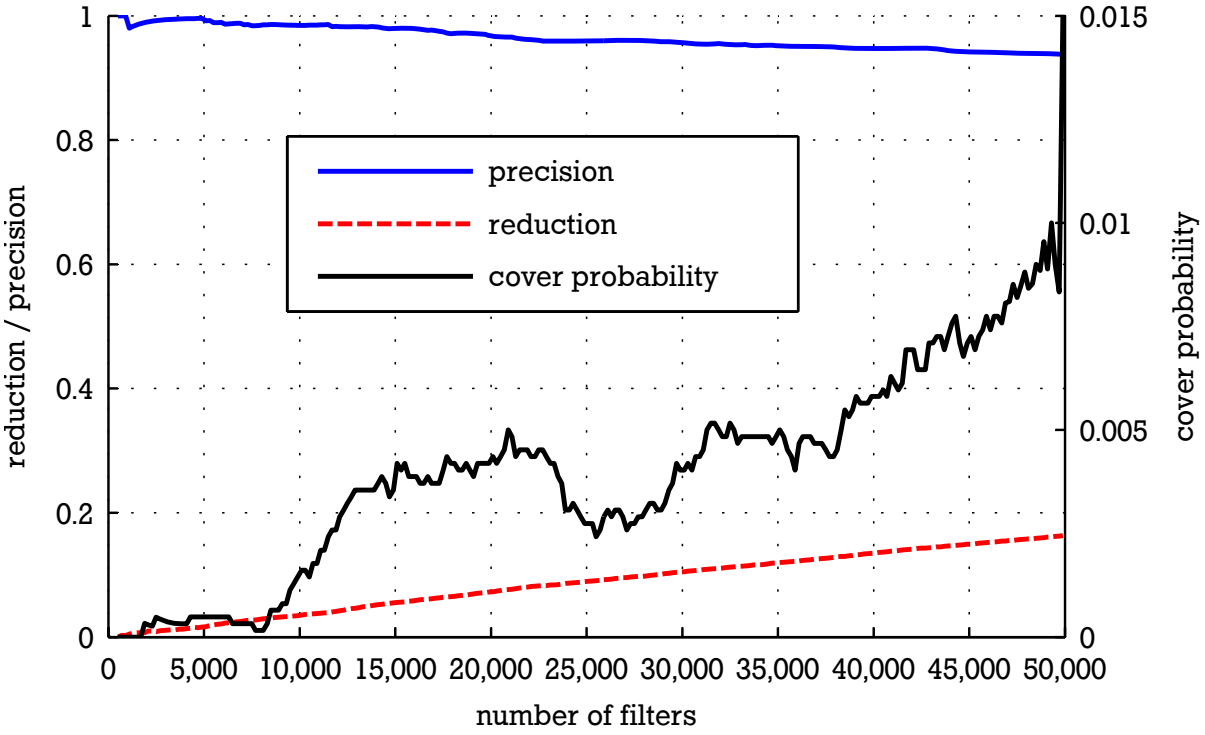
B.2.4 Direct Comparison of Strategies

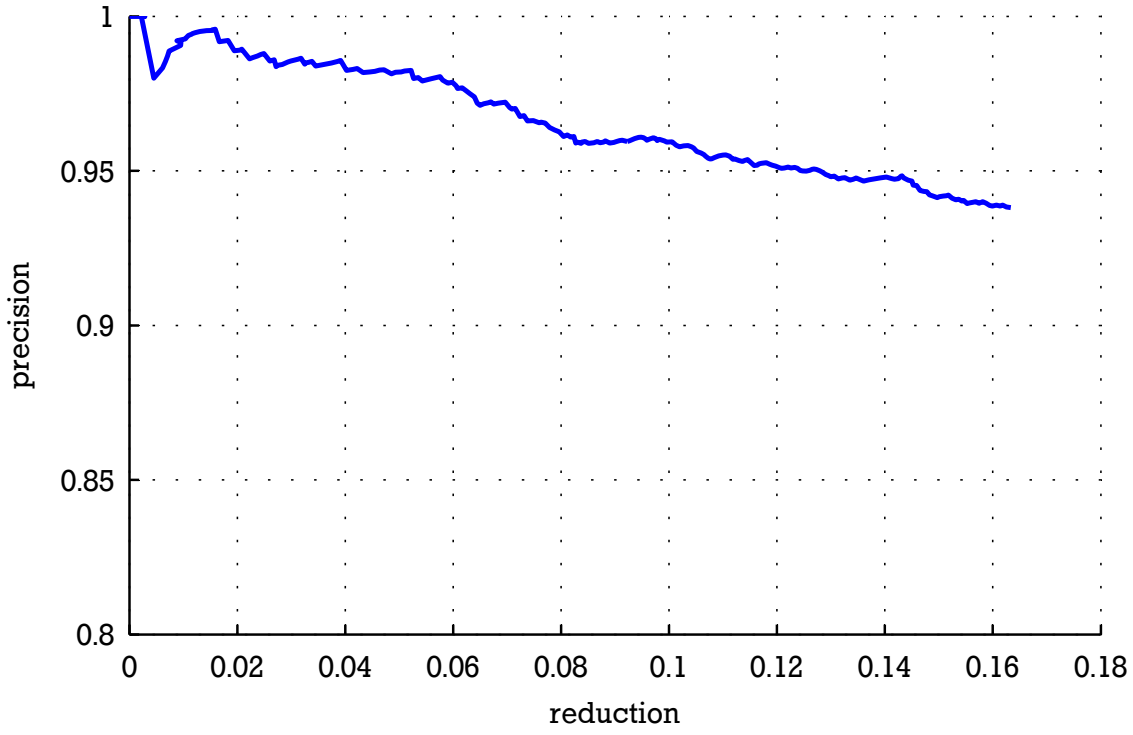
The following figure shows plots for the different strategies. The filters were created from set $\mathscr{R}_6$, and the notifications from set $\mathscr{R}_4$. The sets were chosen such that the plots are typical for the strategies applied to 2-interval filters and notifications with equal characteristics in each dimension. The figure allows a direct comparison of the different merging strategies.



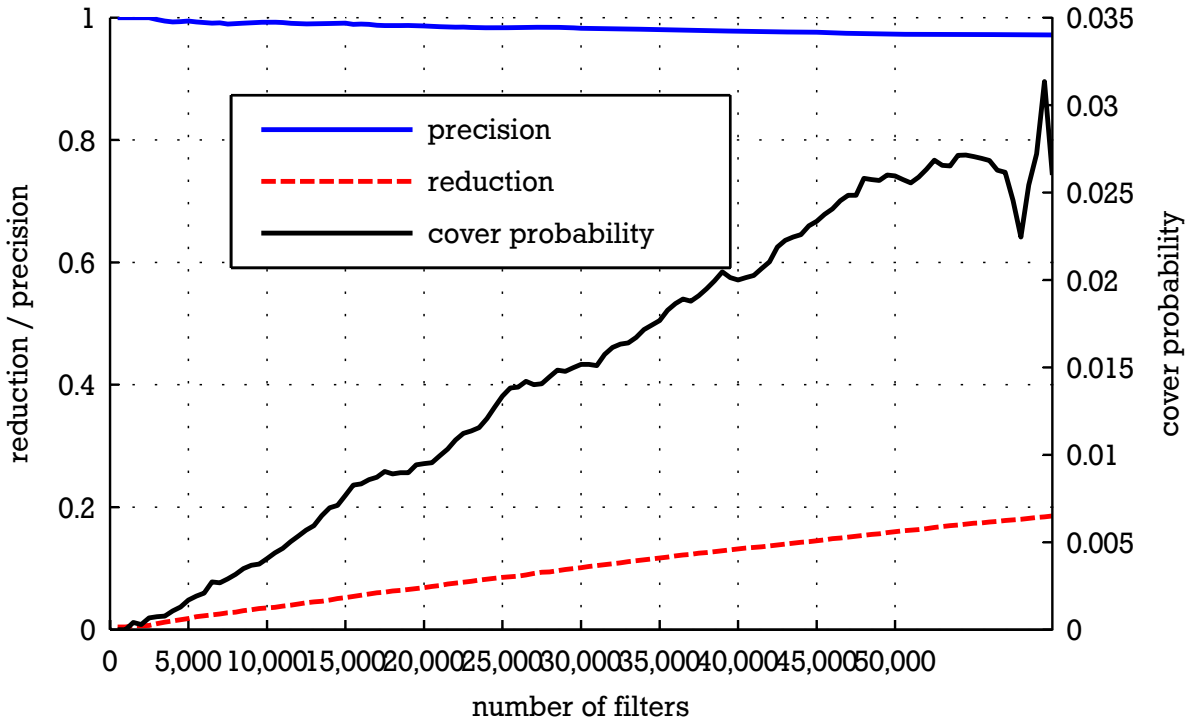
Exemplary direct comparison of size penalty *vs.* mean size penalty *vs.* distance penalty with filters from 2-interval set $\mathscr{R}_6$ and notifications from 2-interval set $\mathscr{R}_4$

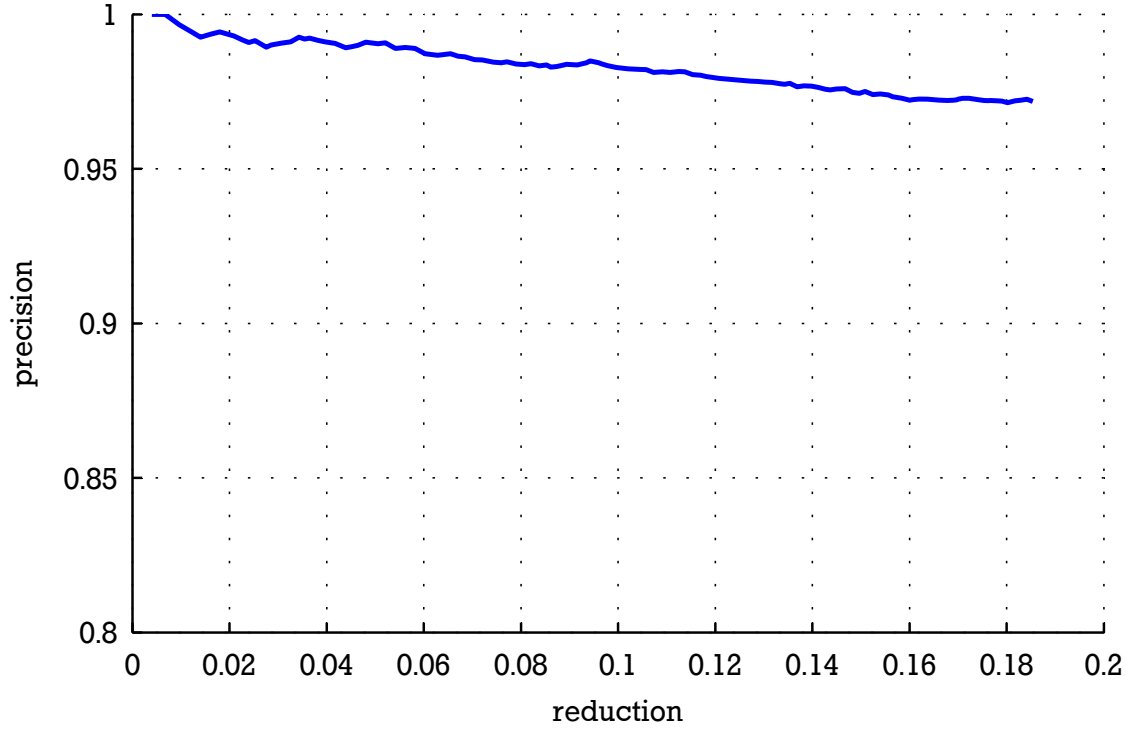
B.2.5 Impact of Filter Set Size



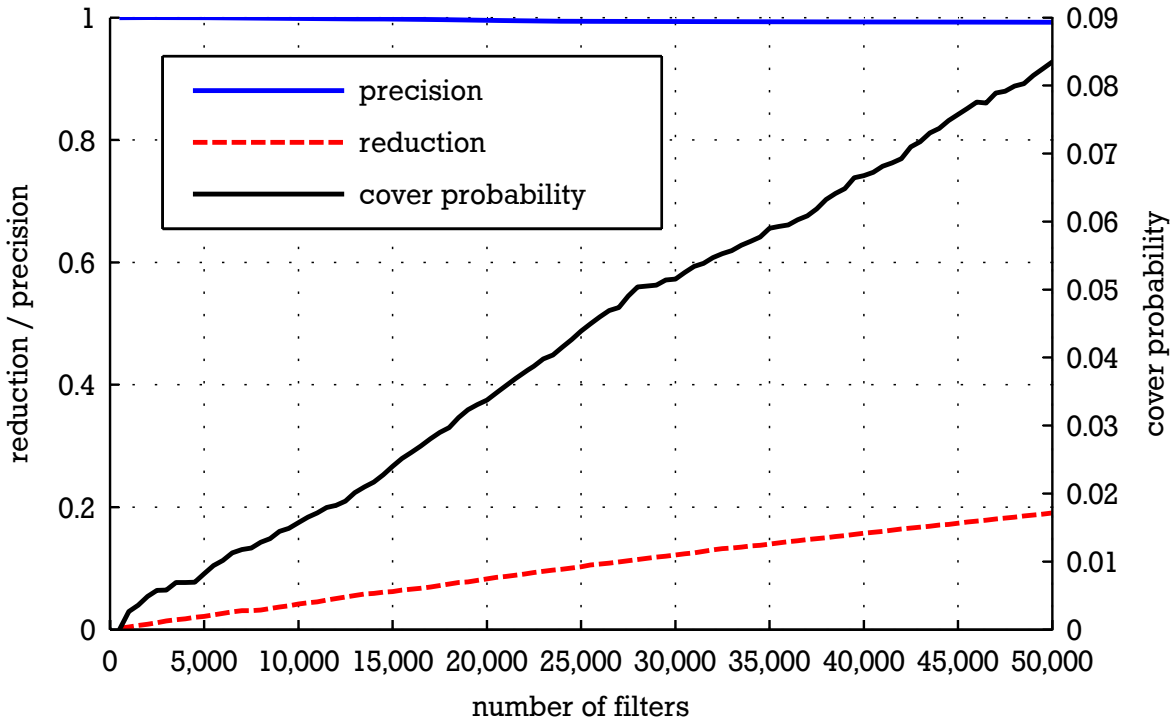


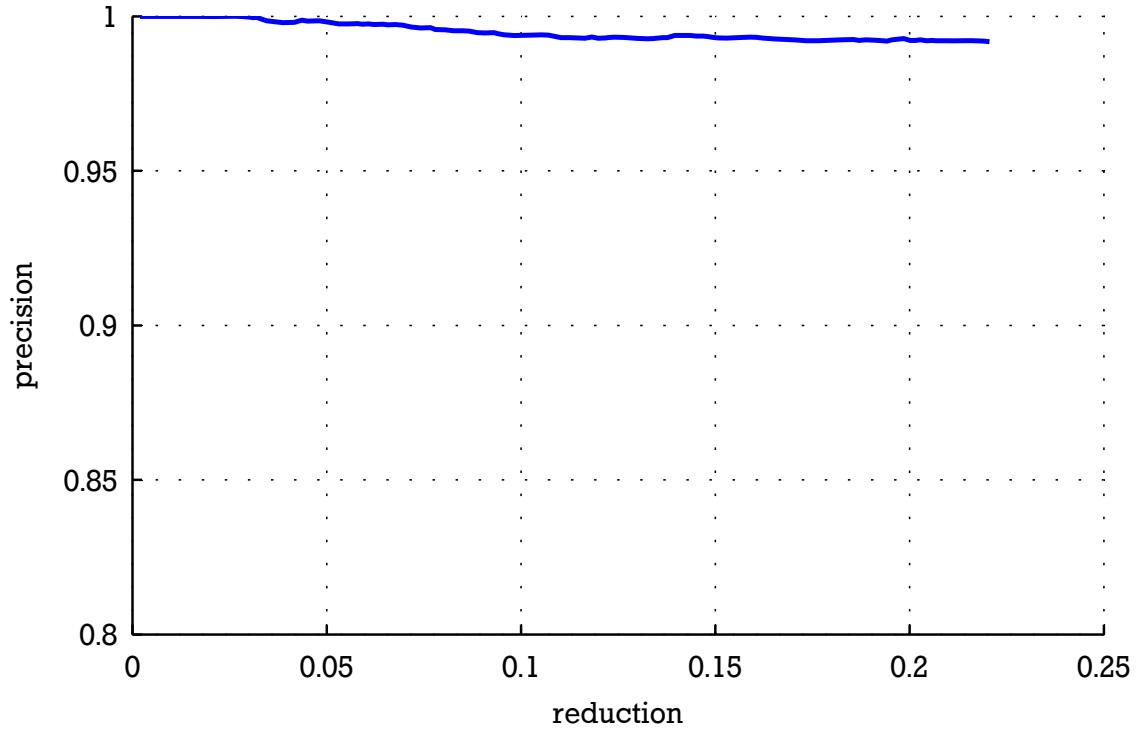
Regular distance penalty score ($p_0 = 2000$) with filters and notifications from 2-interval set $\mathscr{R}_1$



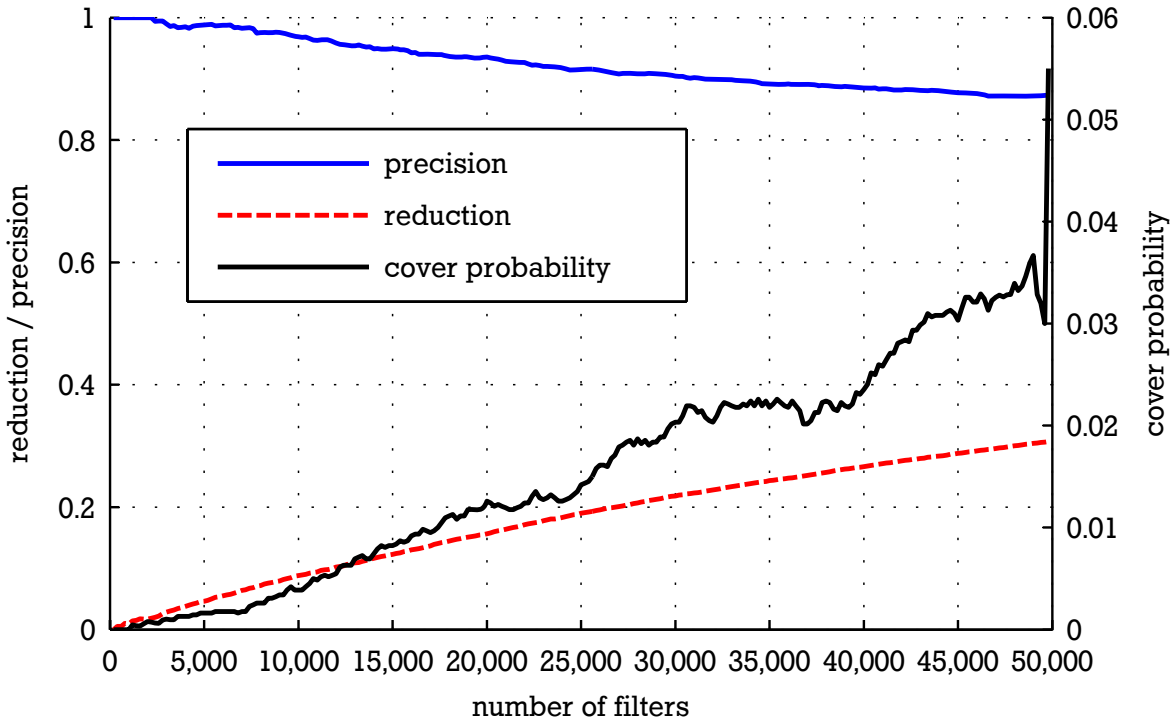


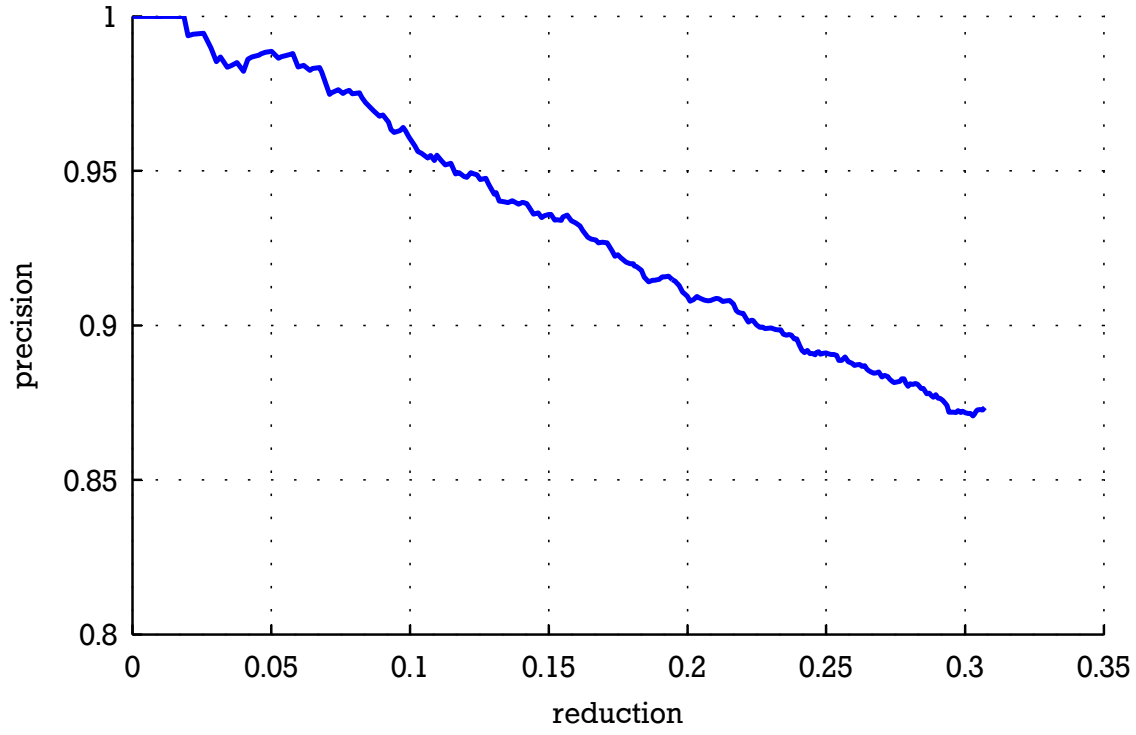
Regular distance penalty score ($p_0 = 2000$) with filters and notifications from 2-interval set $\mathscr{R}_2$



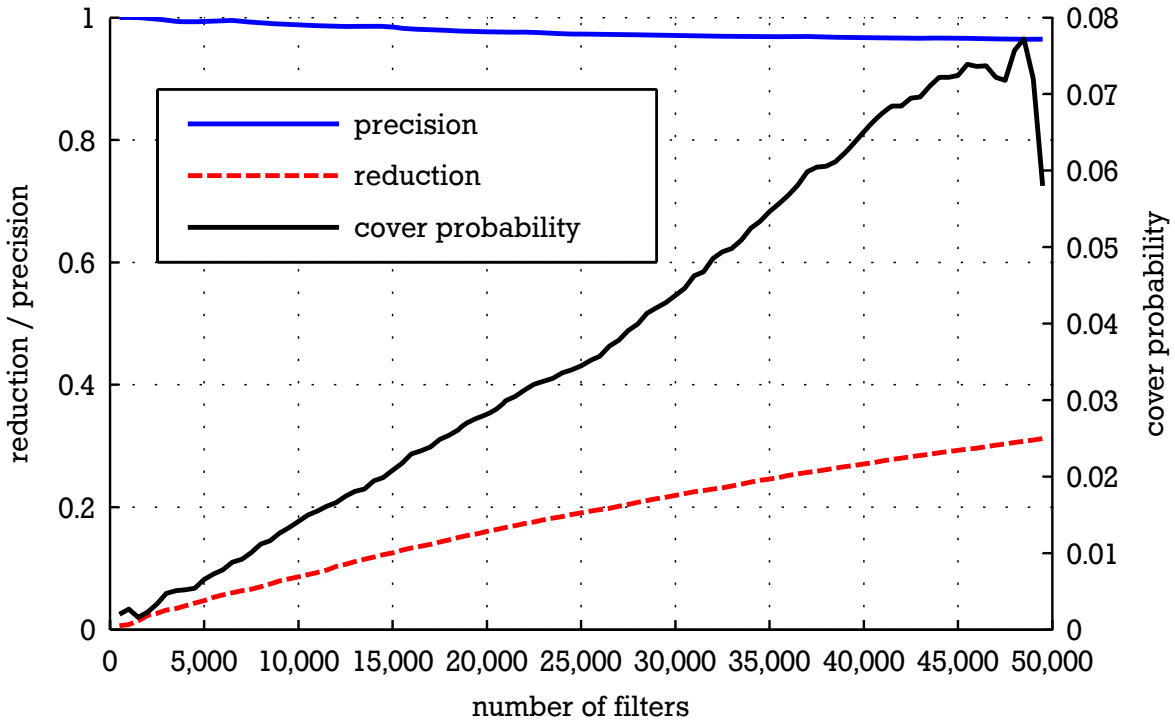


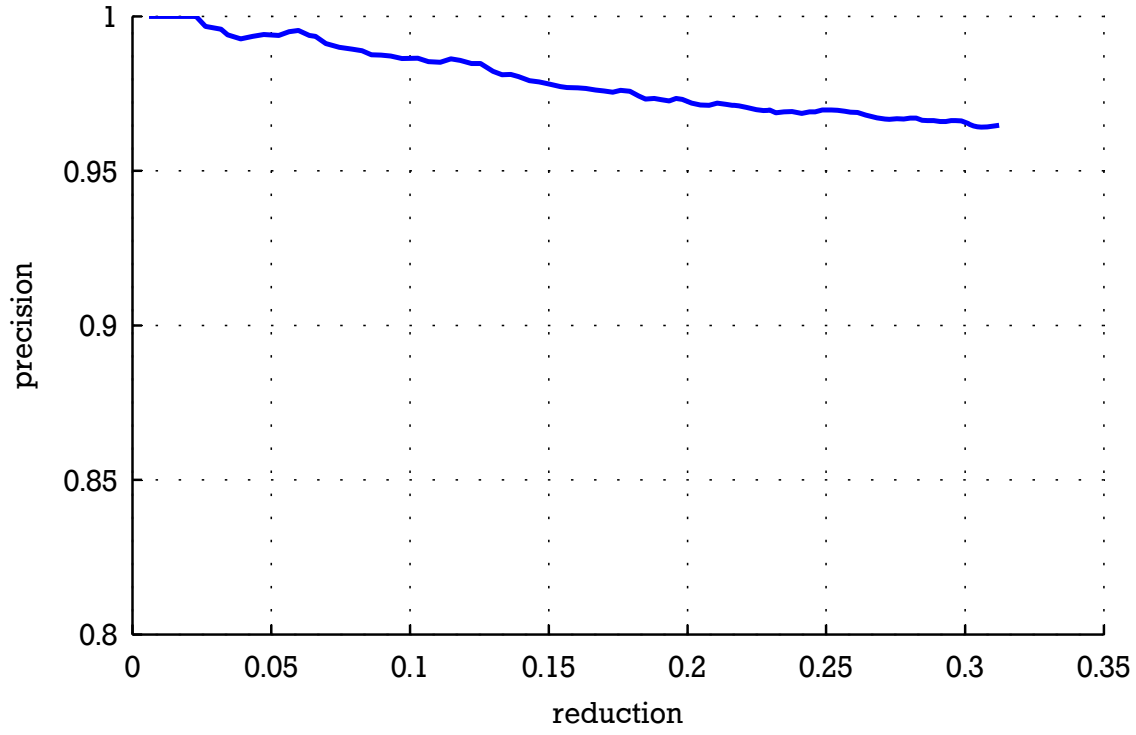
Regular distance penalty score ($p_0 = 2000$) with filters and notifications from 2-interval set $\mathscr{R}_3$



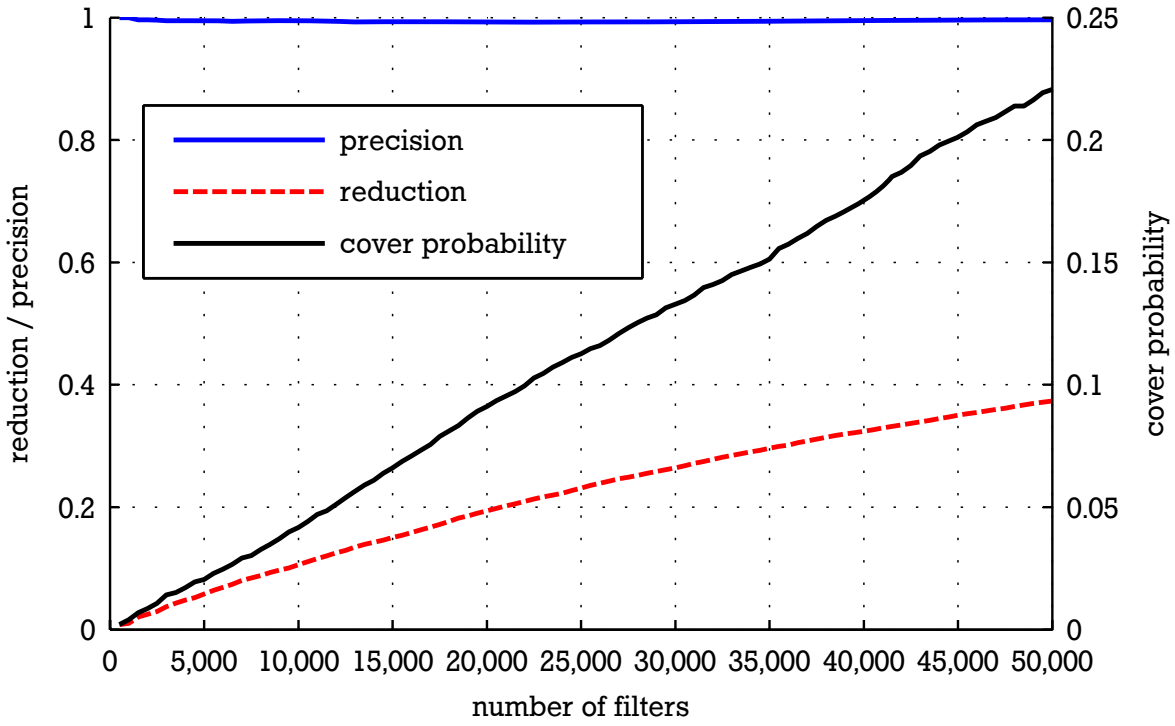


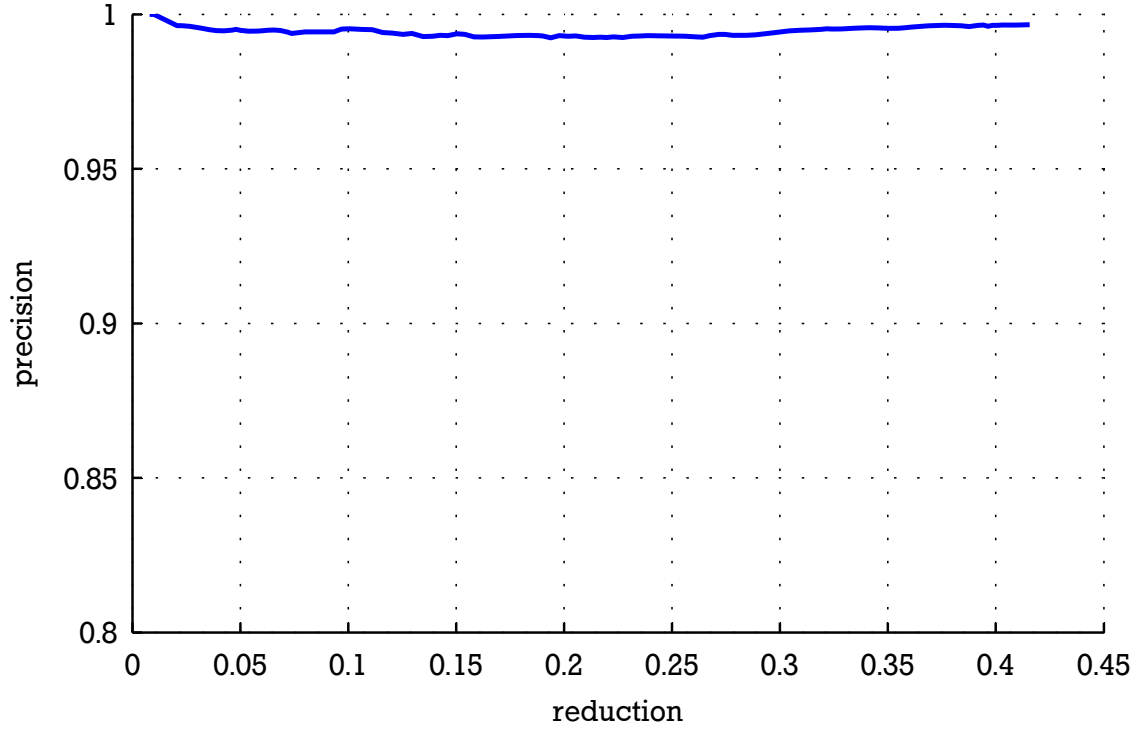
Regular distance penalty score ($p_0 = 2000$) with filters and notifications from 2-interval set $\mathscr{R}_4$





Regular distance penalty score ($p_0 = 2000$) with filters and notifications from 2-interval set $\mathscr{R}_5$





Regular distance penalty score ($p_0 = 2000$) with filters and notifications from 2-interval set $\mathscr{R}_6$

B.3 2-Interval Sets with Different Dimension Characteristics

Seven sets $\mathscr{R}_{11}, \ldots, \mathscr{R}_{17}$ of 2-dimensional integer intervals $R = \left(\left[r_1^-, r_1^+\right), \left[r_2^-, r_2^+\right)\right)$ with different dimension characteristics (relative distance and size of intervals) were generated, and experiments were carried out on these sets using as alternatives for the merging penalty score function *MP*:

- size penalty score s
- mean size penalty score s_{mean}
- distance penalty score d
- normalized distance penalty score d_{norm}

The non-exhaustive merging candidate search was used.

B.3.1 Sample Set Composition

The sets of 2-intervals exhibit widely varying characteristics, different interval sizes in the two dimensions ($\sigma_{|I|}$ between 2^8 and 2^{15}) and different value ranges (between $\left[0, 2^{20}\right]$ and $\left[0, 2^{25}\right]$), which result in different mean distances. The sets were primarily intended to assess the normalized distance-based approach. Every set $\mathscr{R}_{12}, \ldots, \mathscr{R}_{17}$ is a variation of $\mathscr{R}_{11}$ with either different mean size or different mean distance. This allowed to test the approach with 2-interval notifications with different characteristics than 2-interval filters, as described in Section 5.4.4. The intervals in both dimensions of all sets are uniformly distributed in the value range.

Table B.3.: Synthetic sample sets of 2-intervals with different characteristics in each dimension

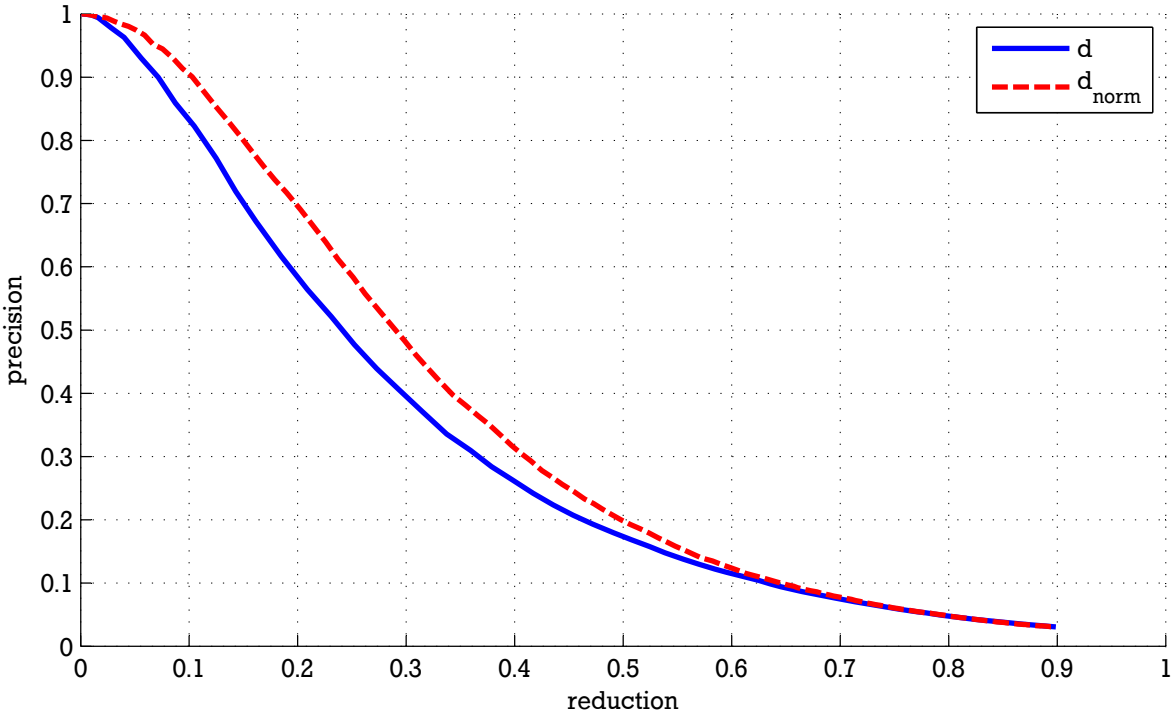
	range dim 1 $[0,\ldots]$	$\sigma_{\lvert I\rvert_1}$	range dim 2 $[0,\ldots]$	$\sigma_{\lvert I\rvert_2}$	difference to $\mathscr{R}_{11}$
$\mathscr{R}_{11}$	2^{23}	2^{13}	2^{20}	2^{10}	
$\mathscr{R}_{12}$	2^{23}	2^{13}	2^{22}	2^{10}	$\overline{d}_2$ larger
$\mathscr{R}_{13}$	2^{25}	2^{13}	2^{20}	2^{10}	$\overline{d}_1$ larger
$\mathscr{R}_{14}$	2^{23}	2^{15}	2^{20}	2^{10}	$\overline{s}_1$ larger
$\mathscr{R}_{15}$	2^{23}	2^{13}	2^{20}	2^{12}	$\overline{s}_2$ larger
$\mathscr{R}_{16}$	2^{23}	2^{11}	2^{20}	2^{10}	$\overline{s}_1$ smaller
$\mathscr{R}_{17}$	2^{23}	2^{13}	2^{20}	2^{8}	$\overline{s}_2$ smaller

In the following, results for the distance-based penalty score functions are presented in direct comparison of the two variants, the regular distance penalty score, and the

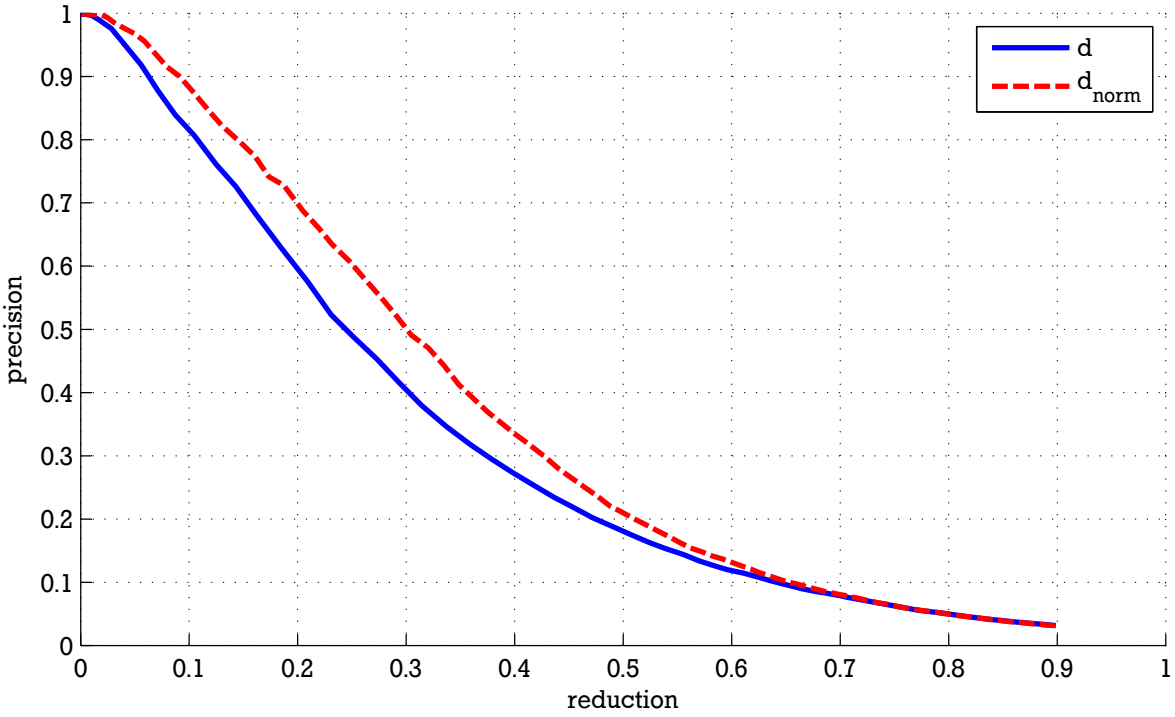
normalized distance penalty score. The results of the size-based penalty score functions presented subsequently are supplied for completeness only. The superiority of both distance-based penalty score functions was shown before. It is again confirmed by these results.

Note that the figures in the following are different from the ones above in that the results from two different strategies using the same sets of filters and notifications are plotted in one figure.

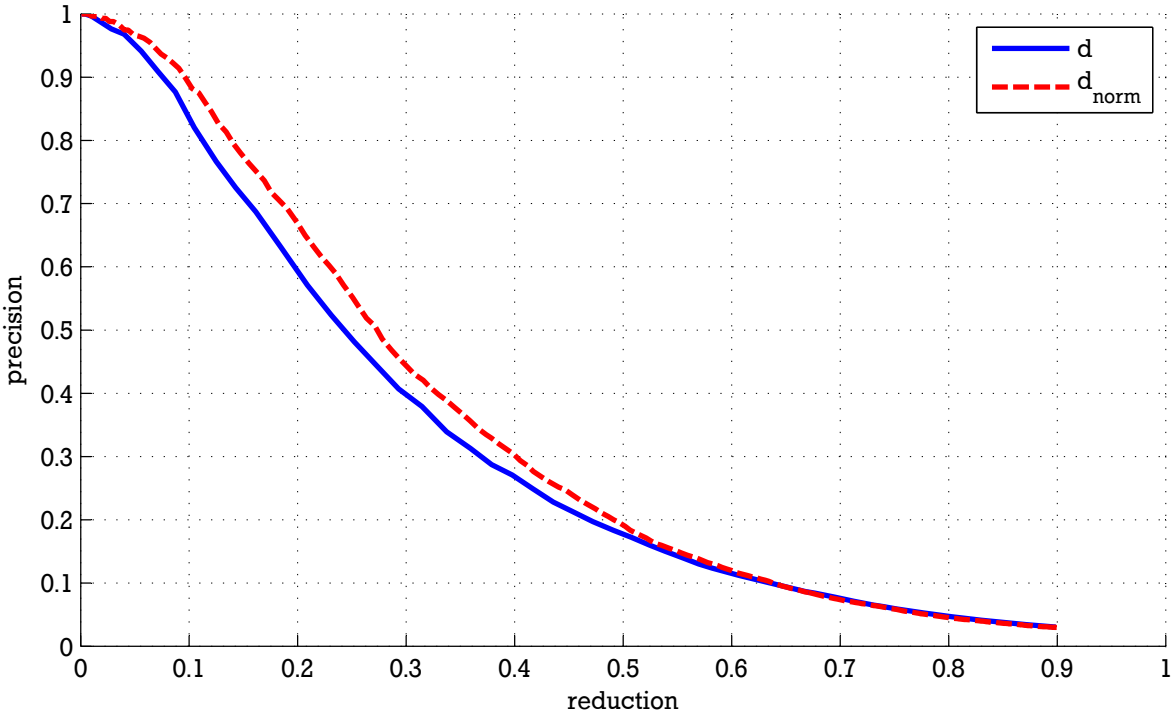
B.3.2 Distance-based Penalty Scores



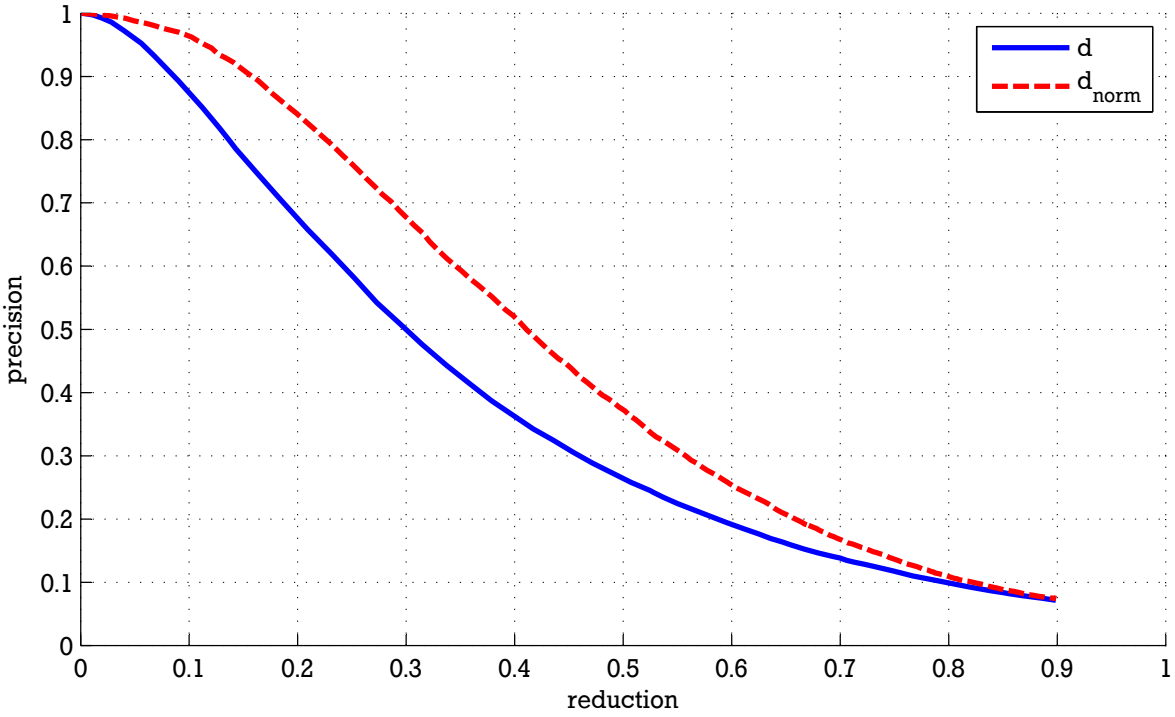
Distance-based penalty scores with filters and notifications from 2-interval set $\mathscr{R}_{11}$



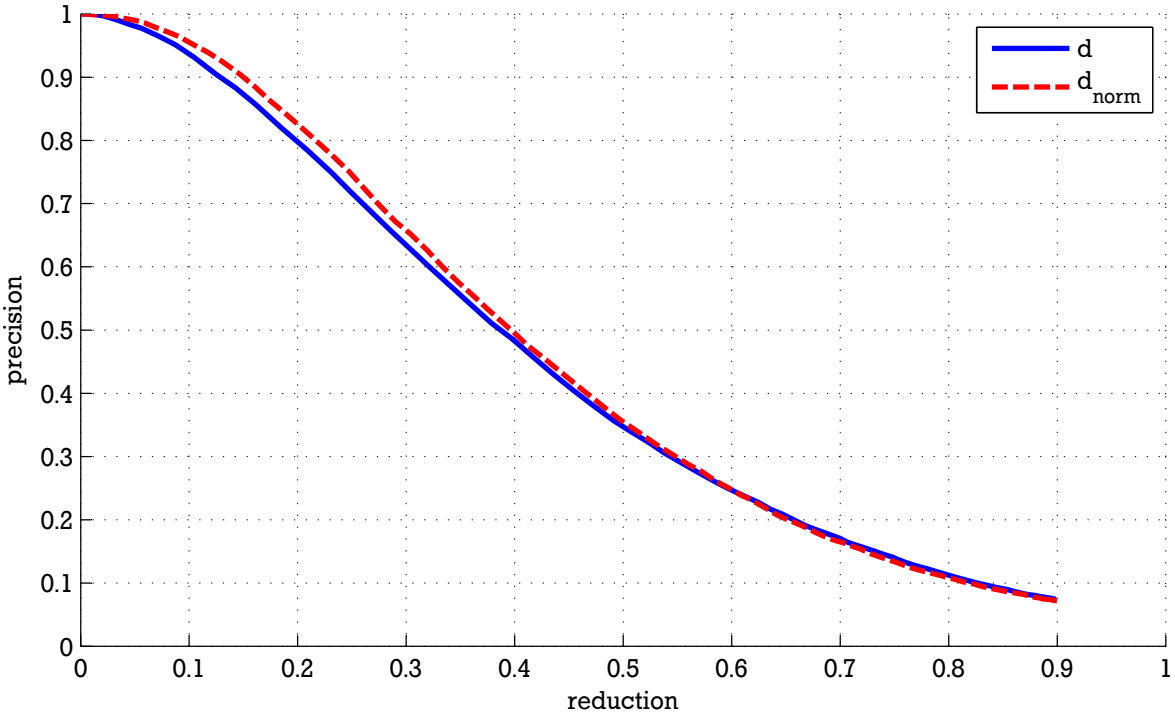
Distance-based penalty scores with filters from set $\mathscr{R}_{11}$ and notifications from $\mathscr{R}_{12}$



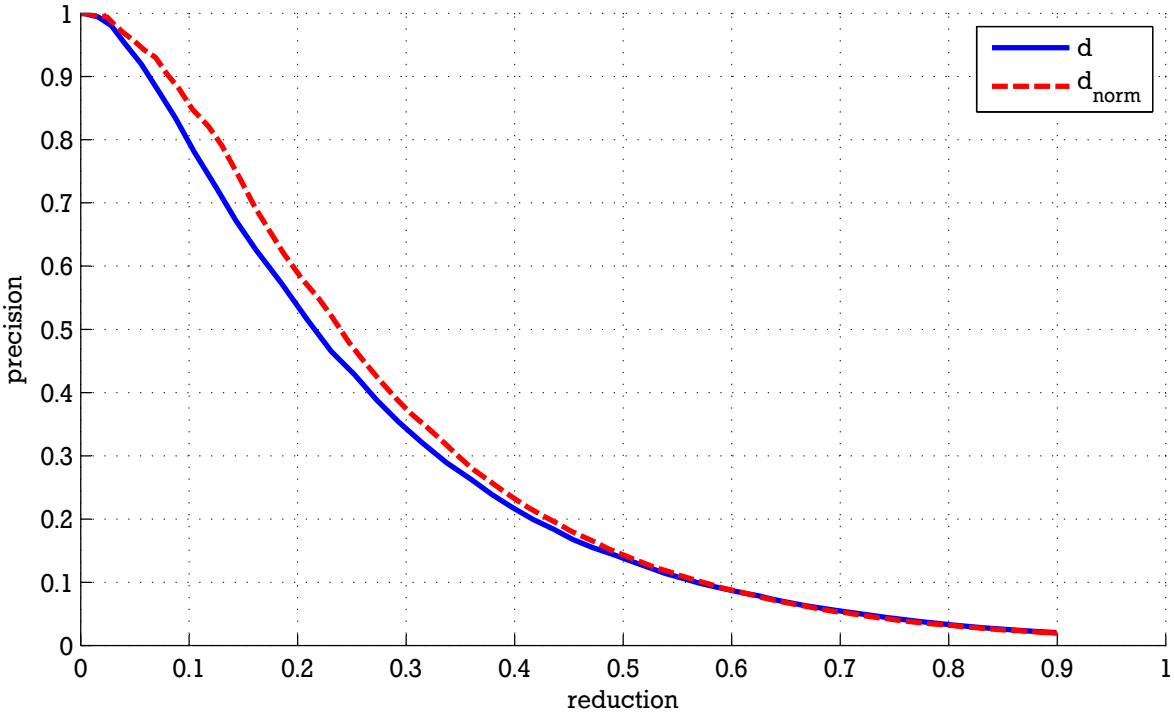
Distance-based penalty scores with filters from set $\mathscr{R}_{11}$ and notifications from $\mathscr{R}_{13}$



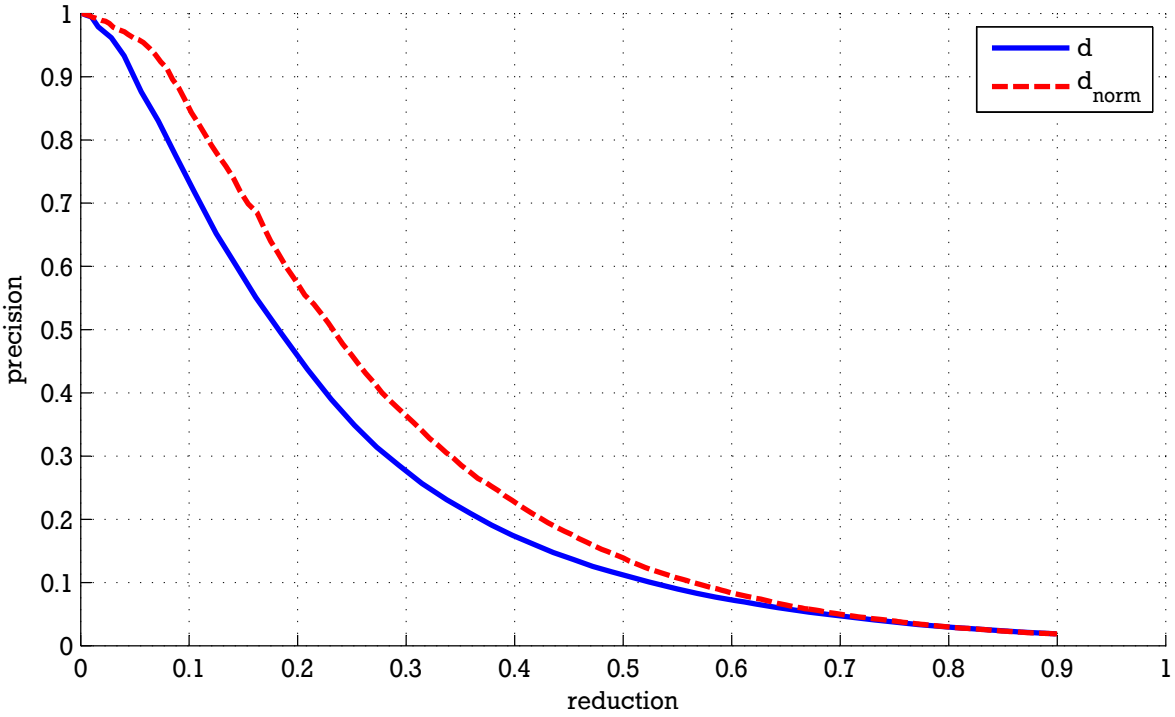
Distance-based penalty scores with filters from set $\mathscr{R}_{11}$ and notifications from $\mathscr{R}_{14}$



Distance-based penalty scores with filters from set $\mathscr{R}_{11}$ and notifications from $\mathscr{R}_{15}$

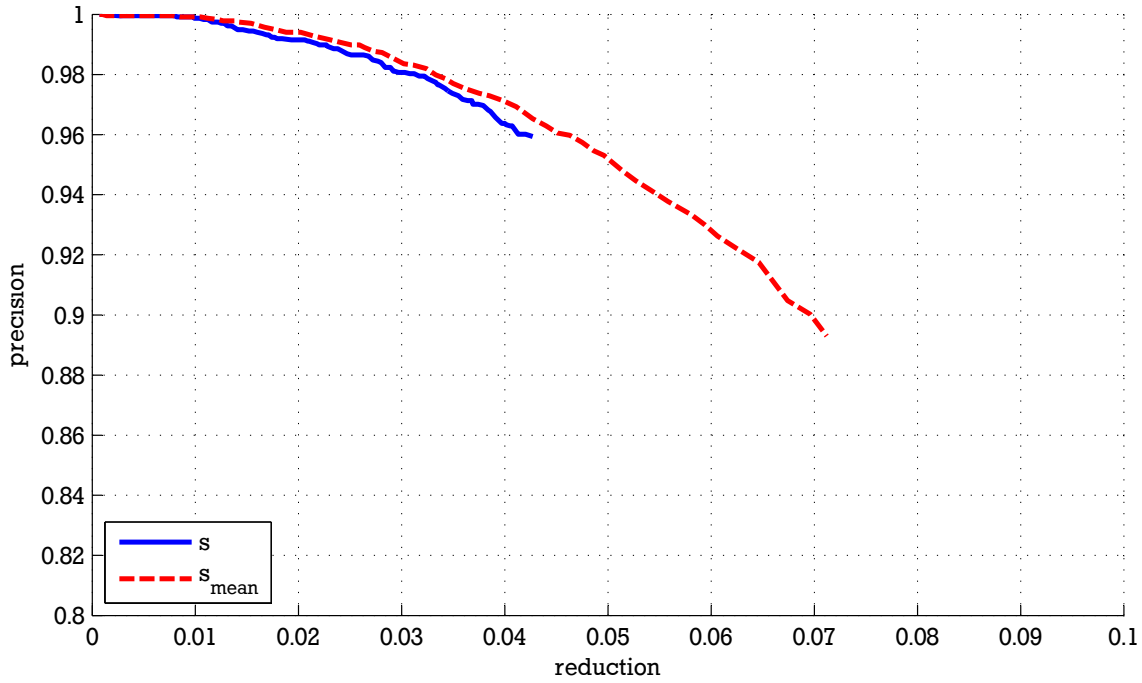


Distance-based penalty scores with filters from set $\mathscr{R}_{11}$ and notifications from $\mathscr{R}_{16}$

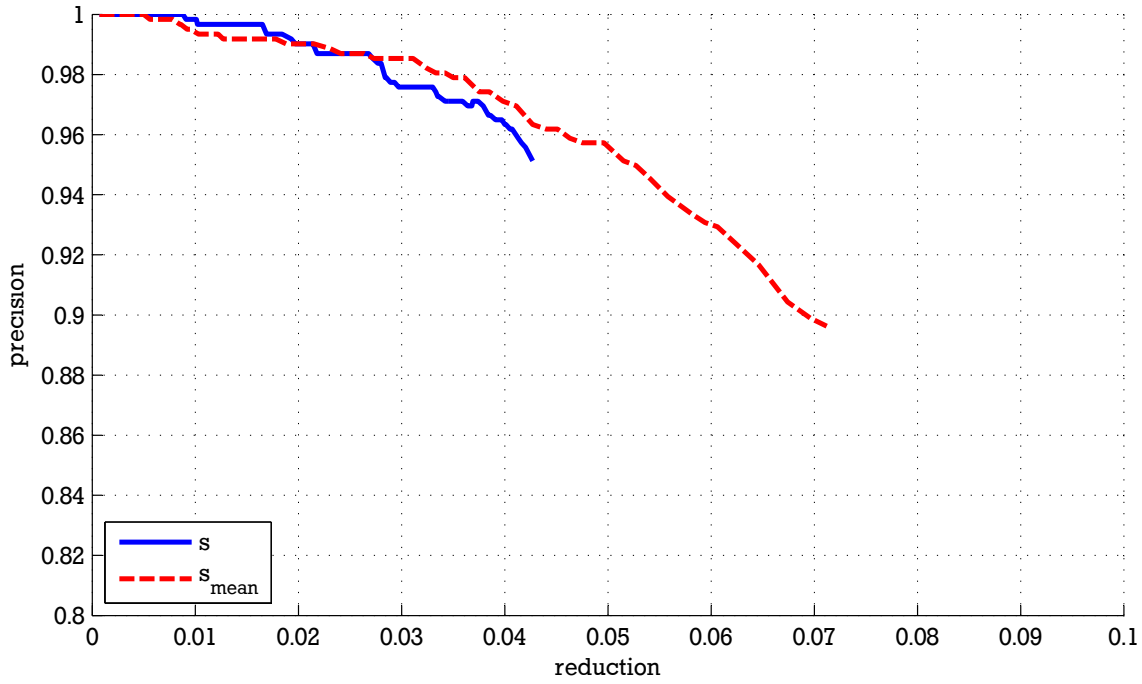


Distance-based penalty scores with filters from set $\mathscr{R}_{11}$ and notifications from $\mathscr{R}_{17}$

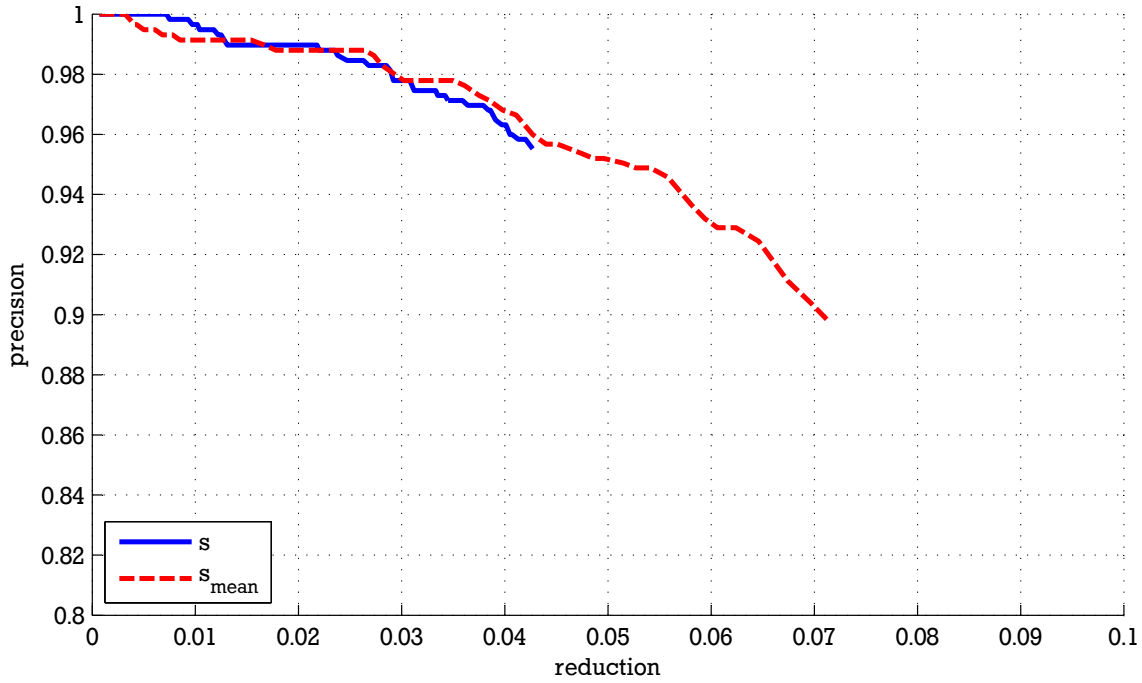
B.3.3 Size-based Penalty Scores



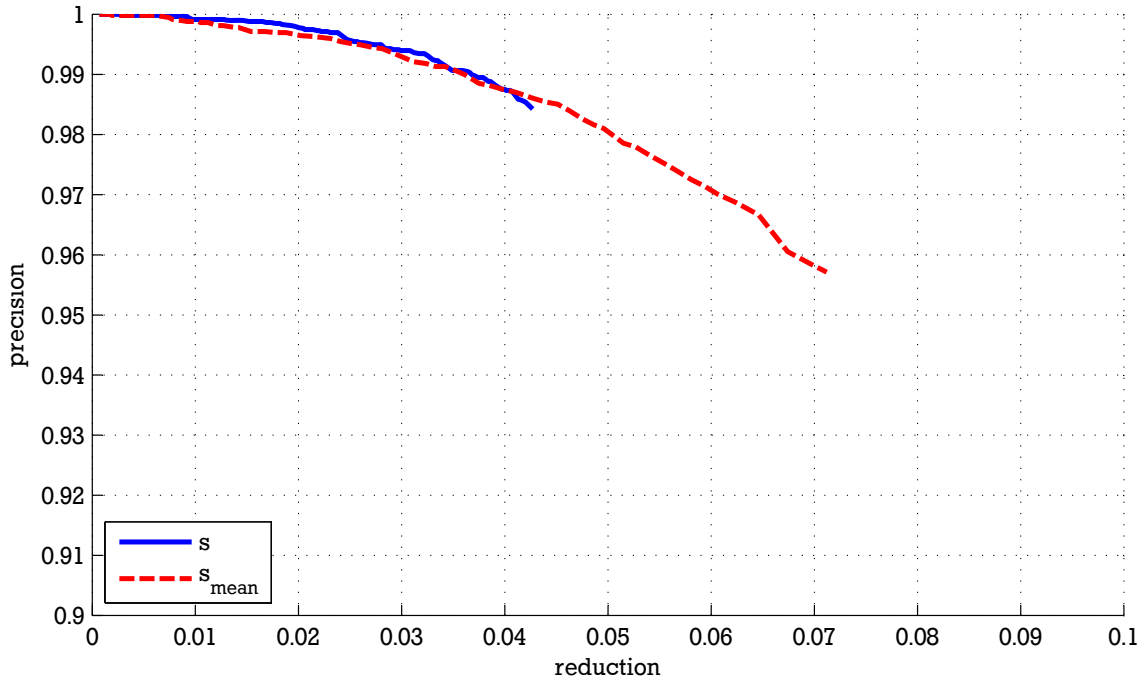
Size-based penalty scores with filters and notifications from set $\mathscr{R}_{11}$



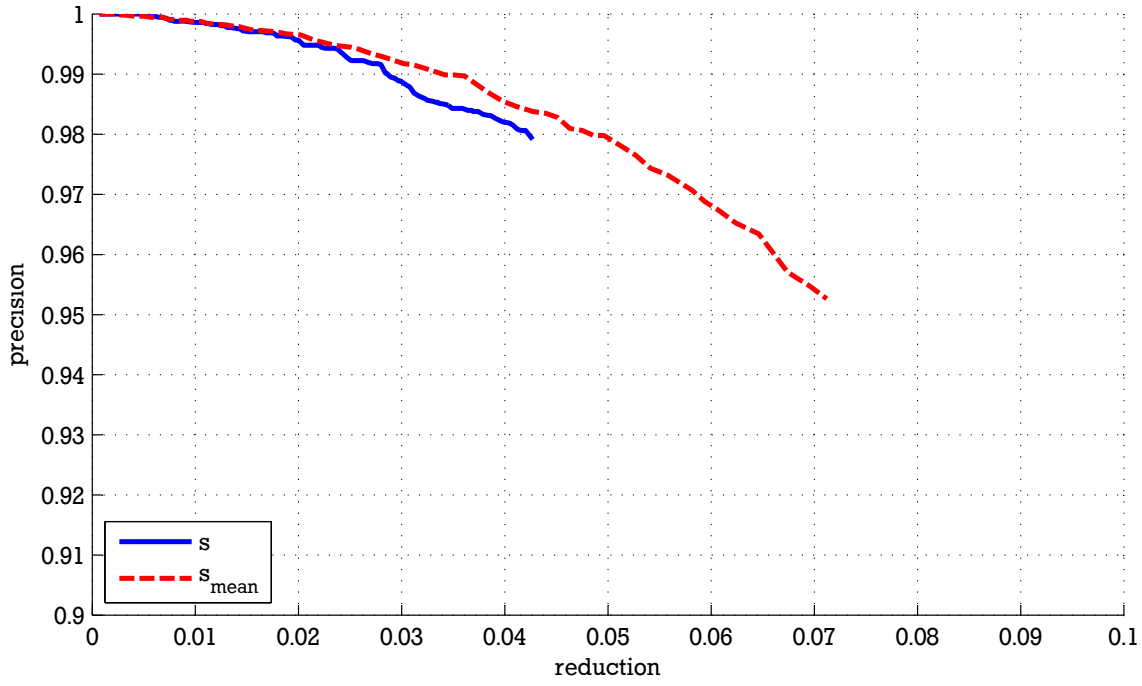
Size-based penalty scores with filters from set $\mathscr{R}_{11}$ and notifications from $\mathscr{R}_{12}$



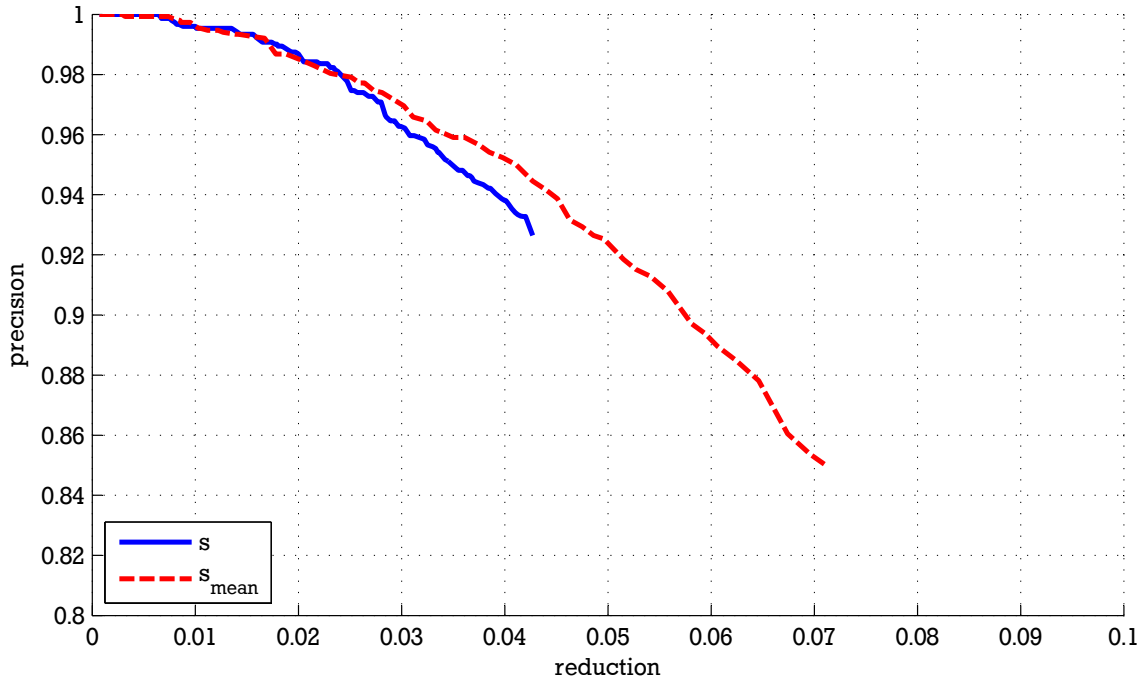
Size-based penalty scores with filters from set $\mathscr{R}_{11}$ and notifications from $\mathscr{R}_{13}$



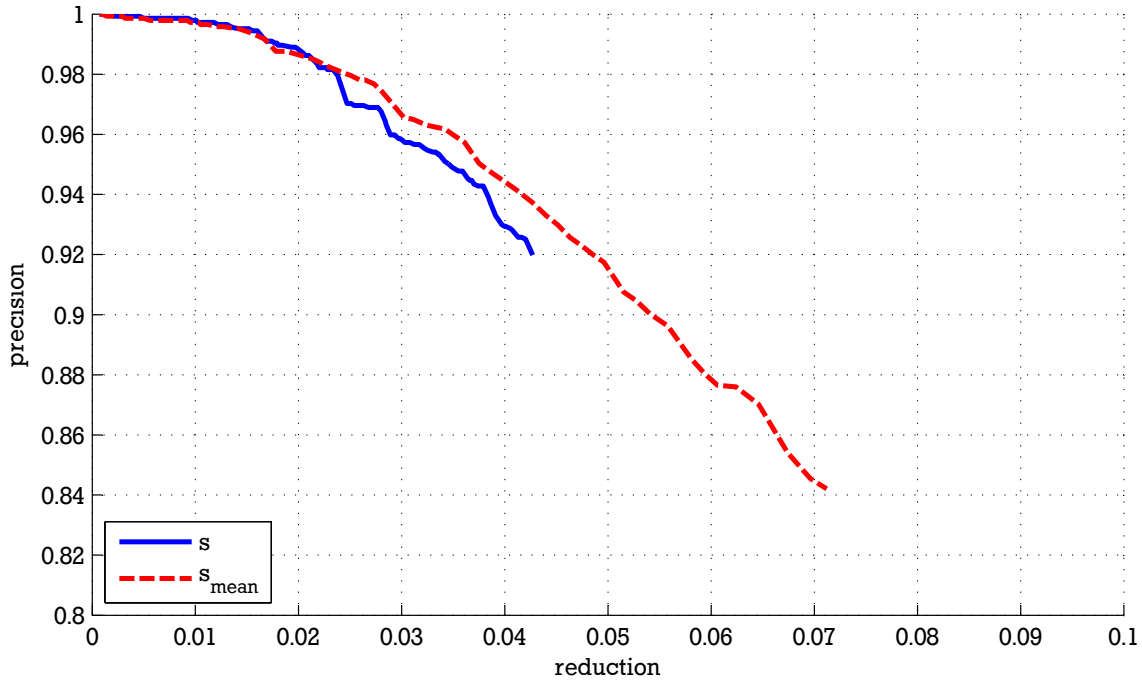
Size-based penalty scores with filters from set $\mathscr{R}_{11}$ and notifications from $\mathscr{R}_{14}$



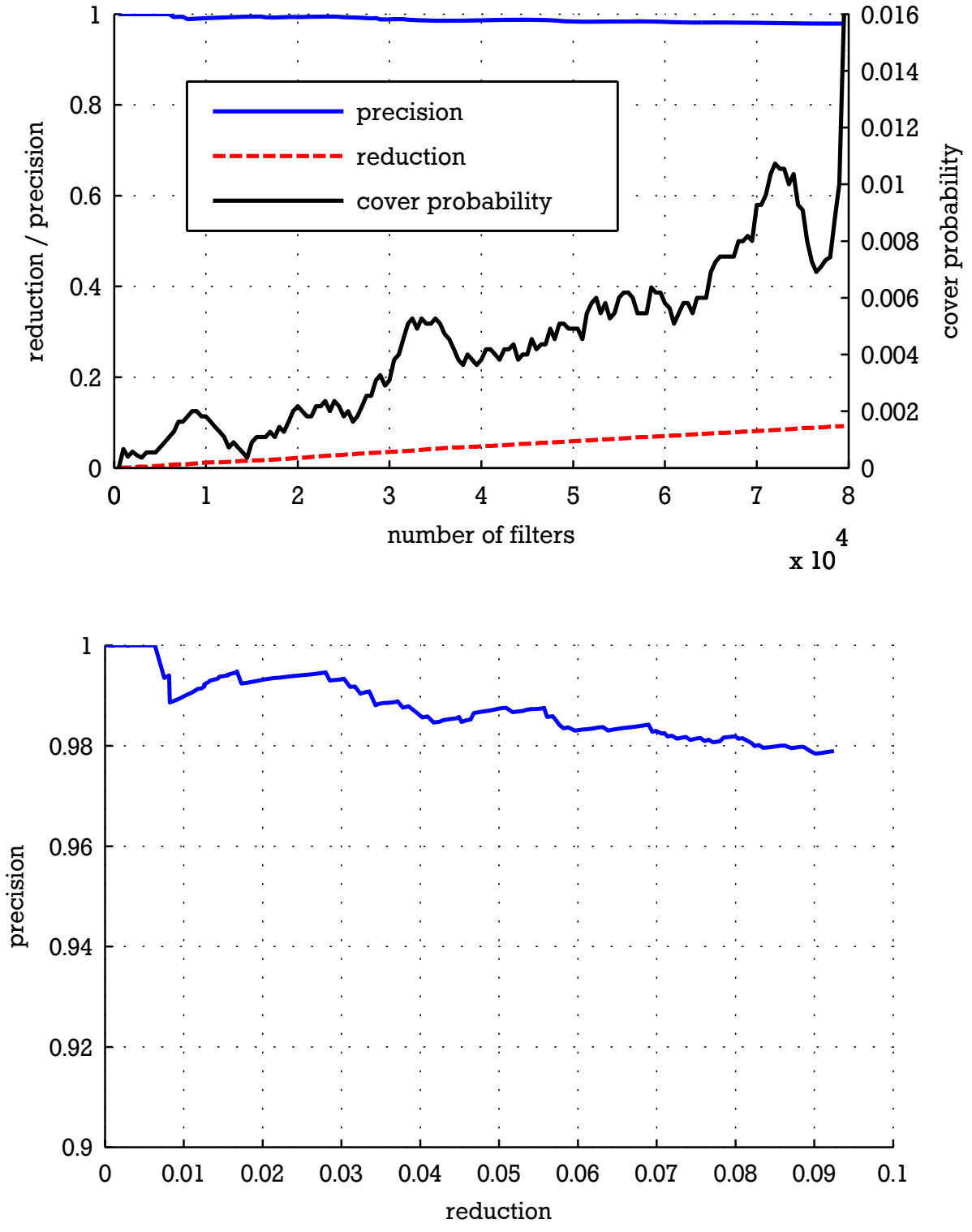
Size-based penalty scores with filters from set $\mathscr{R}_{11}$ and notifications from $\mathscr{R}_{15}$



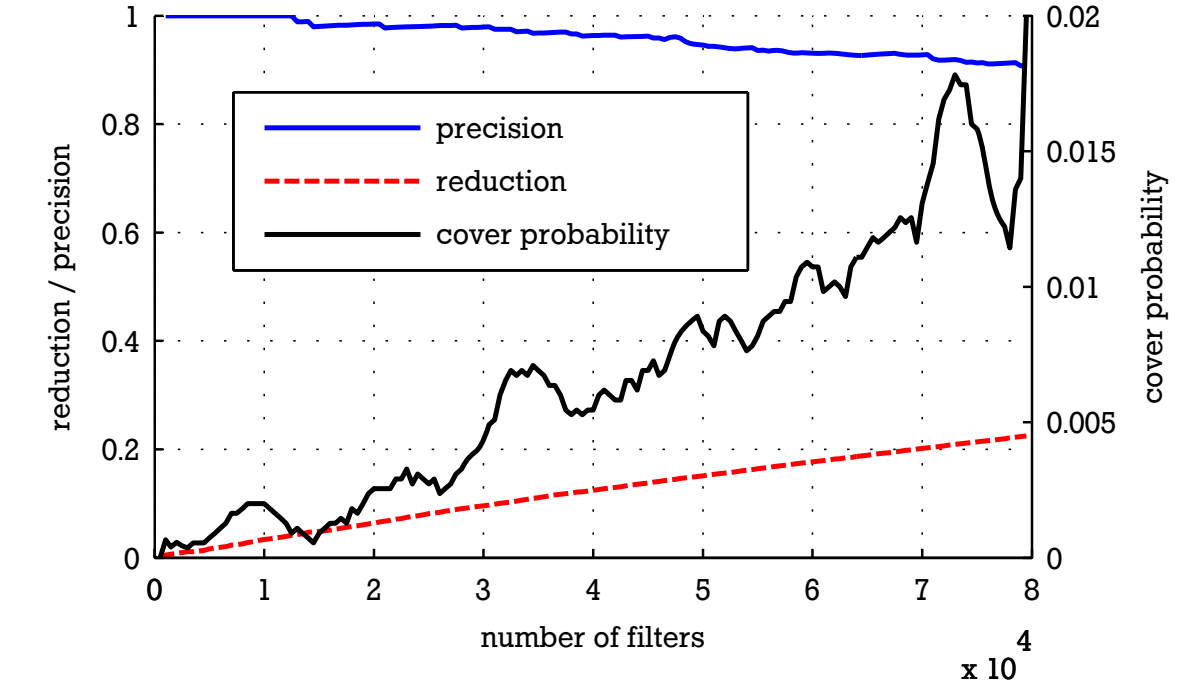
Size-based penalty scores with filters from set $\mathscr{R}_{11}$ and notifications from $\mathscr{R}_{16}$

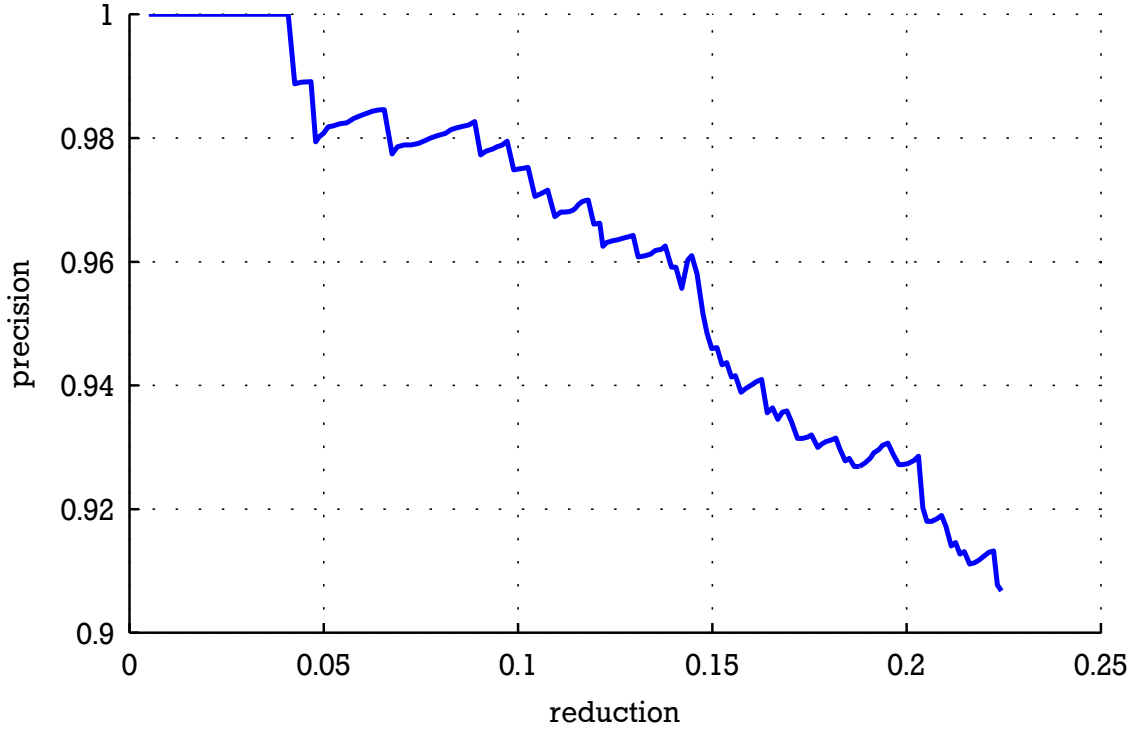


Size-based penalty scores with filters from set $\mathscr{R}_{11}$ and notifications from $\mathscr{R}_{17}$

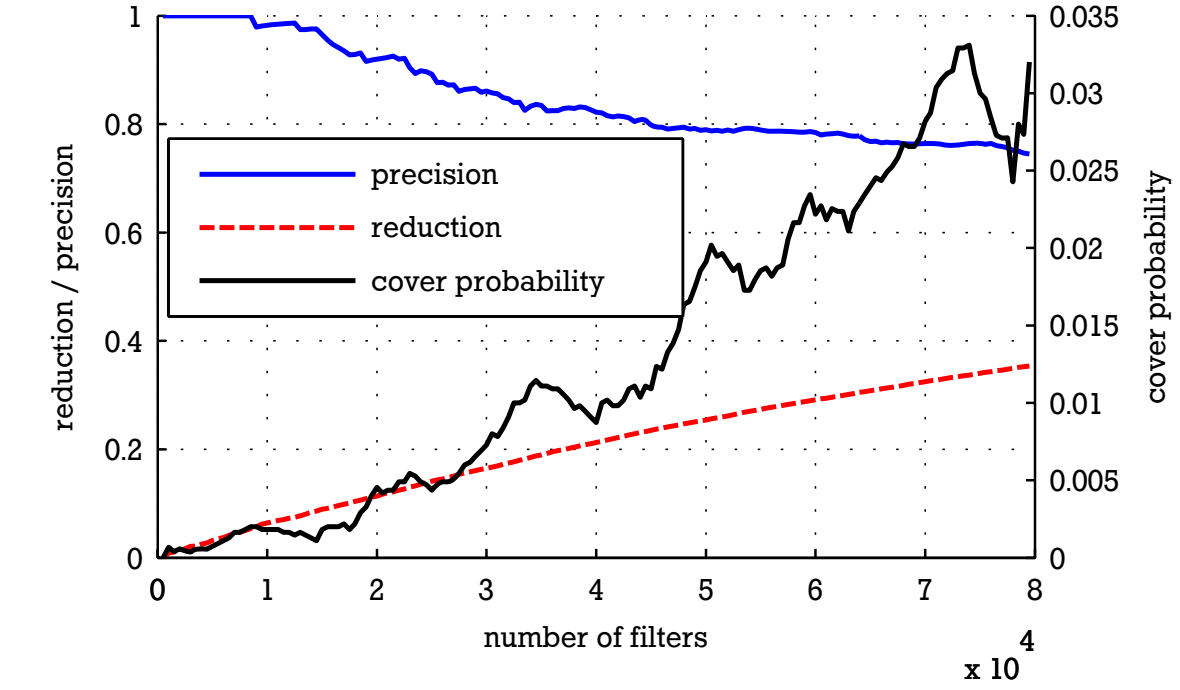


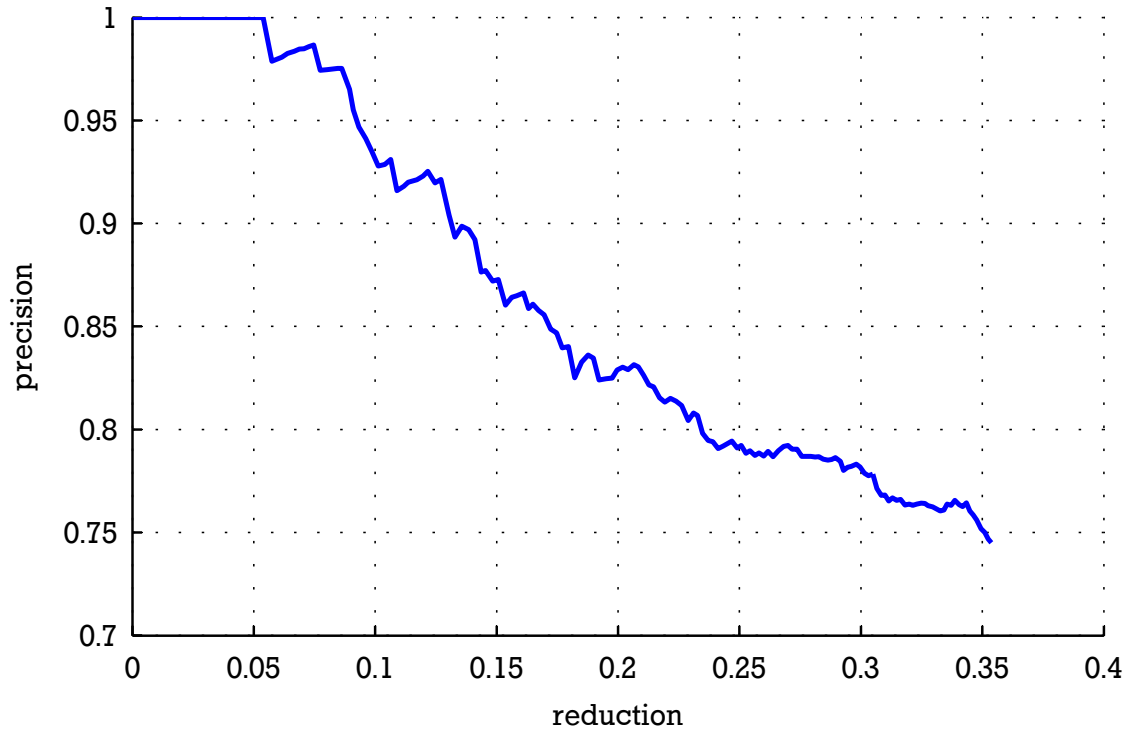
Normalized distance penalty score ($p_0 = 18,840$) with filters and notifications from set $\mathscr{R}_{11}$





Normalized distance penalty score ($p_0 = 25,140$) with filters from set $\mathscr{R}_{11}$ and notifications from $\mathscr{R}_{12}$





Normalized distance penalty score ($p_0 = 69,160$) with filters from set $\mathscr{R}_{11}$ and notifications from $\mathscr{R}_{13}$

C Filter Handling Scheme Experiments

For the experiments described in Section 6.4 in which we investigated the effects of our filter handling scheme, we designed a test scenario focused on a day in the European airspace in the year 2022, and created a large set of appropriate scenario data. Section C.1 details how we created realistic flight data for this future day by extrapolating real historic flight data that was provided by EUROCONTROL in the frame of the AIXM 5 research project.

Due to lack of real aeronautical event data, we generated synthetic spatiotemporal notifications for aeronautical events. However, we aimed to create the data with characteristics as realistic as possible for the application scenario. The generation of this experimental event data is described in Section C.2. Complete experimental results are detailed in Section C.3.

C.1 Flight Trajectory Subscriptions

Regulations in Europe require to file a flight plan for every planned IFR flight in advance that has to be confirmed by the authorities. Such a flight plan contains, besides basic information like departure, arrival, and alternate airport, estimated time en route, aircraft equipment, pilot's name and number of people on board etc., the intended maximum flight level and the planned route of the flight along the aeronautical infrastructure, typically taking the form <departure airport>–<waypoint>–<airway>–<waypoint>–<airway>– ⋯ –<waypoint>–<arrival airport>.[1] This flight plan is used by the authorities for ATM planning and to initiate tracking and routing services. A coarse representation of the calculated, deconflicted trajectory (so-called *model-1 profile*) is confirmed to the operator.

The CFMU inside EUROCONTROL keeps a record of every IFR flight passing through European airspace. Such a record consists of the flight plan information, the model-1 profile, and (for past flights) the *model-3 profile*: a representation of the actually flown trajectory as surveiled by radar. The CFMU provided us on request with an anonymized[2] data set of such records of all 58,492 IFR flights that were conducted in European airspace on August 5 and 6, 2007.[3]

[1] Waypoints are named points in airspace used for airspace organization, e.g., as airway junctions, designed by the airspace authority and published through AIS.

[2] For security and privacy reasons, all information directly linking an individual flight to the operator/airline was stripped out (flight number and all aircraft information: type, category, id, etc.).

[3] These two days were chosen because the air traffic on these days is characteristic for the year: The number of flights per day is approximately the average number in 2007, and no particular irregular-

Both profiles take the form of a temporally ordered sequence of point records P denoting a latitude/longitude waypoint (ϕ, λ) with an associated timestamp t and an altitude a given as flight level. The profiles of the 2007 flight data served as the basis for the trajectory subscriptions for our experiments.

As scenario day, we chose August 5, 2022, and temporally moved all profiles from both days in 2007 to this day, thus approximately doubling the number of flights per day to account for the expected increase in air traffic volume. We assumed that the future Business Trajectory is a much more precise representation of the flight trajectory than the profiles in the data set and therefore linearly interpolated each leg of the profiles with three segments, such that the number of points in each trajectory grows from N in the profile to $3N+1$ in the synthetic Business Trajectory. The trajectory of a flight thus contains 70 legs on average. The subscription for one flight thus consists of a set $\mathscr{F}$ of spatiotemporal filters for the individual legs with $|\mathscr{F}| \approx 70$.

Each trajectory point takes the form $P = (\phi, \lambda, t, a)$. A spatiotemporal flight subscription was generated by creating individual spatiotemporal filters $F = (H_F, V_F, T_F)$ for each interpolated leg of the trajectory between two successive points $P_1 = (\phi_1, \lambda_1, a_1, t_1), P_2 = (\phi_2, \lambda_2, a_2, t_2)$, as follows:

- The 2D horizontal spatial filter region H is created from the linestring $\ell s = \big((\phi_1, \lambda_1), (\phi_2, \lambda_2)\big)$ by adding a buffer of 5 nm resulting in a polygon $H_F = \text{buffer}(\ell s, 5\,\text{nm})$.[4].

- The vertical interval V is created as $V = \big[\min(a_1, a_2), \max(a_1, a_2)+1\big)$. Hence, the altitude dimension of the filter space is implemented in discrete steps of 1 flight level.

- The temporal interval T is given by $T = \big[t_1, t_2+1\big)$, where chronons with a duration of 1 millisecond was used. Hence, t_1 and t_2 are timestamps of a millisecond precision.

C.2 Aeronautical Event Notifications

We generated synthetic spatiotemporal event notifications $n = (H_E, V_E, T_E)$ for the test scenario with the following characteristics:

ities (as they appear for instance for Christmas traffic) were observed at those days. Furthermore, a Friday and a Saturday was chosen to account for different air traffic characteristics on weekdays and weekends.

[4] The common geometric buffer operation was used: The filter polygon is the result of the Minkowski sum of the linestring ℓs with a circle with 5nm radius.

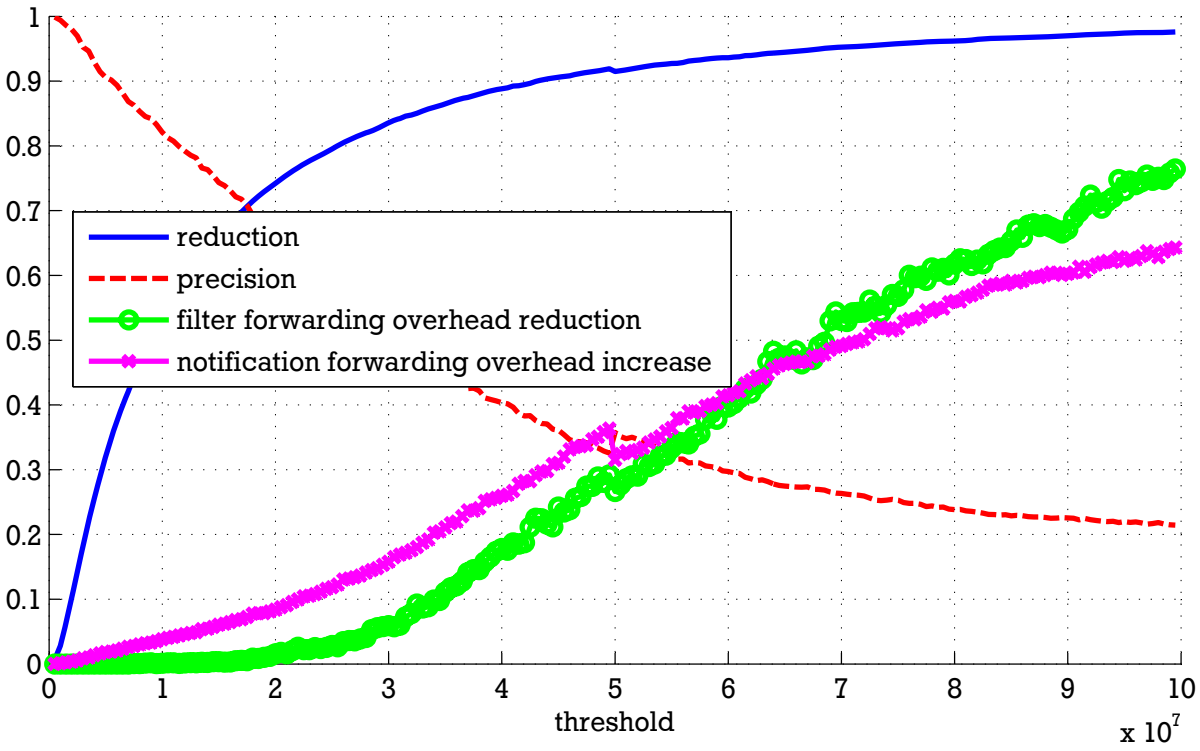
- As horizonal spatial effectivity of the events, we generated random simple polygons of different shapes with 4 to 20 vertices in sizes such that the diameter follows a half-normal distribution with $\sigma = 15\,\text{nm}$ shifted by $+0.5\,\text{nm}$. These polygons were rotated by a random angle, and distributed in a bivariate normal distribution with μ at the geographic center of Europe $\lambda = 8°, \sigma = 47°$, and with standard deviation 8° in north-south direction and 6° in east-west direction.
- For the vertical interval $V_E = \left[v^-, v^+\right)$, a lower flight level v^- and an upper flight level v^+ was chosen such that $v^- \in [0, 349]$ in a half-normal distribution with $\sigma = 100$ and $v^+ \in [v^- + 1, 401]$ in a uniform distribution.
- We assumed that the number of aeronautical events effective at a given point in time roughly correlates with the amount of air traffic, and therefore generated the temporal effectivity interval $T_E = \left[t^-, t^+\right)$ such that there are more events effective in the afternoon hours: t^- was chosen in a normal distribution with μ equal to August 2022, 12:00:00 UTC (2 p.m. Central European Summer Time) and $\sigma = 3\,\text{hours}$. We assumed one fifth of all aeronautical events to be permanent events with $t^+ = t^- + 1$, the temporal duration $t^+ - t^-$ of the other intervals was chosen in a half-normal distribution with $\sigma = 5\,\text{hours}$.

Table C.1.: Characteristics of aeronautical event notifications

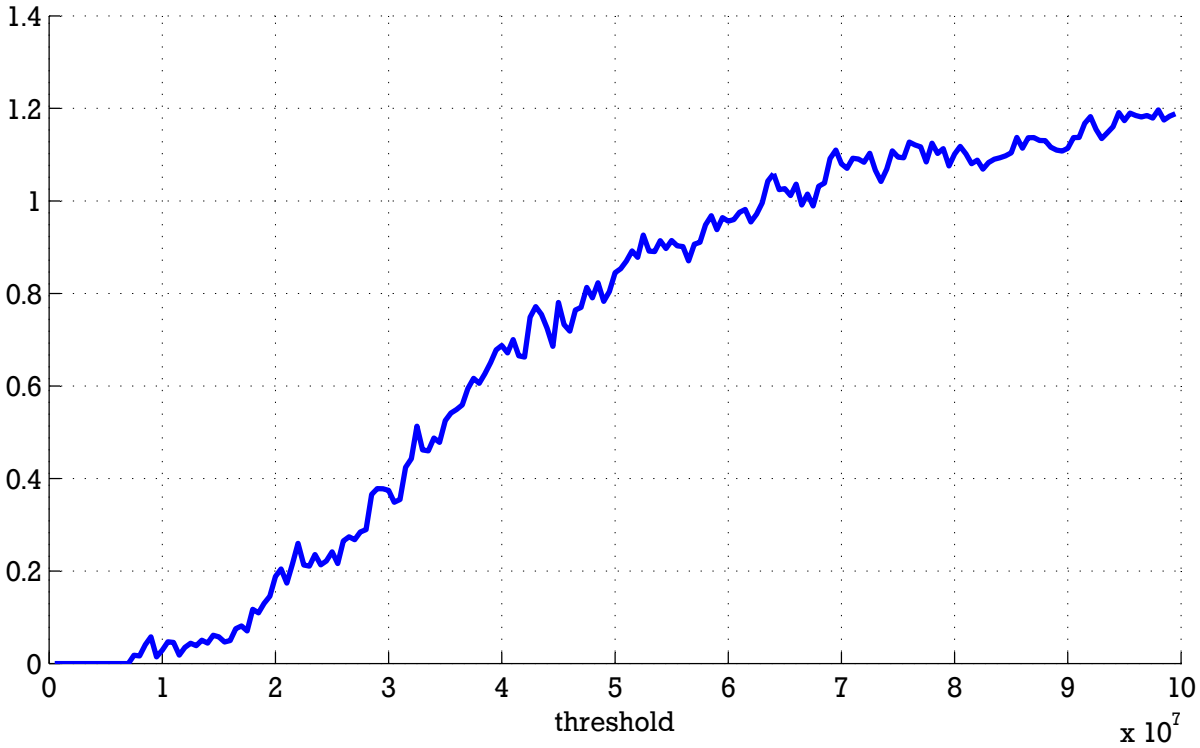
	mean size	mean distance	normalization factor
dimension 1: longitude	1,420	25	4,700
dimension 2: latitude	1,420	14	5,000
dimension 3: flight level	164	0.036	350,000
dimension 4: time	6,083,150	10,910	2

Table C.1 details the characteristics of the notification set. We computed the mean distance and mean size and, based on these, the normalization factors. The two horizontal spatial dimensions exhibit the same mean size, because the polygons that were used for the horizontal spatial region of the event notifications were rotated by a random angle, but different mean distances because of the different standard deviations used for their positioning in longitude and latitude. As expected, the temporal dimension, which uses one millisecond as chronon, exhibits by far the largest mean distance and mean size, and therefore, the other dimensions are stretched.

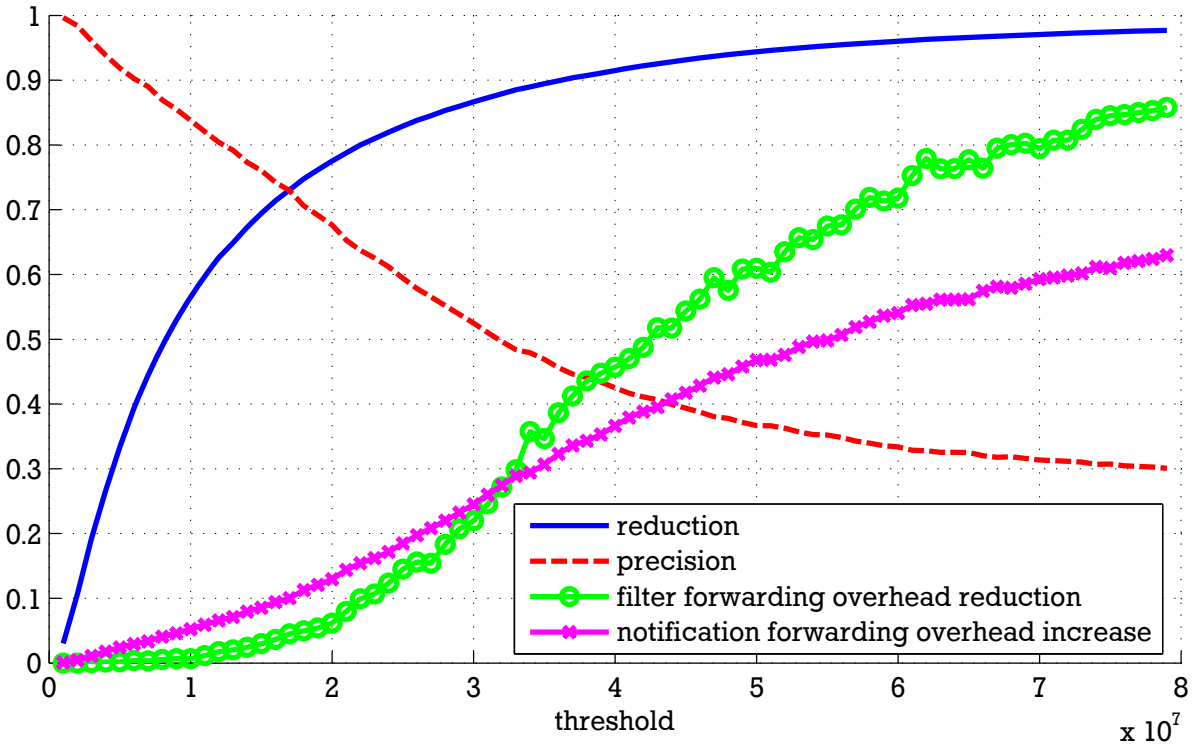
C.3 Complete Experimental Results



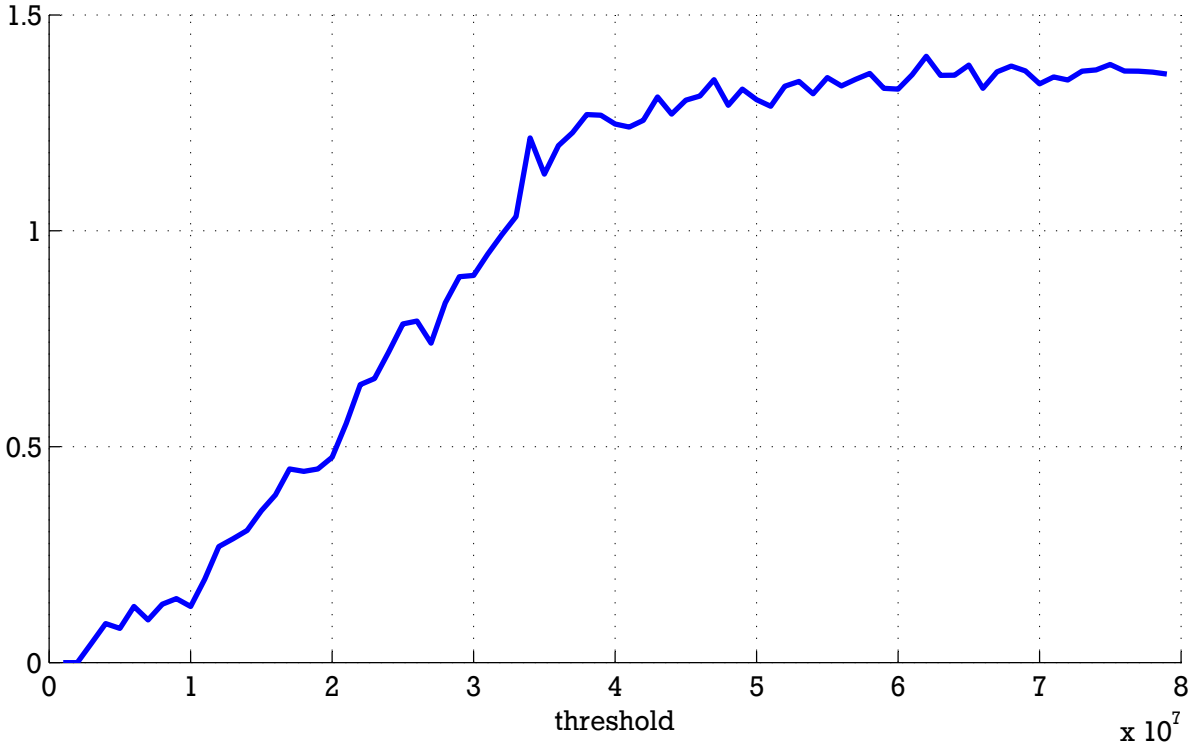
Characteristics of a set of 200 random flight subscriptions



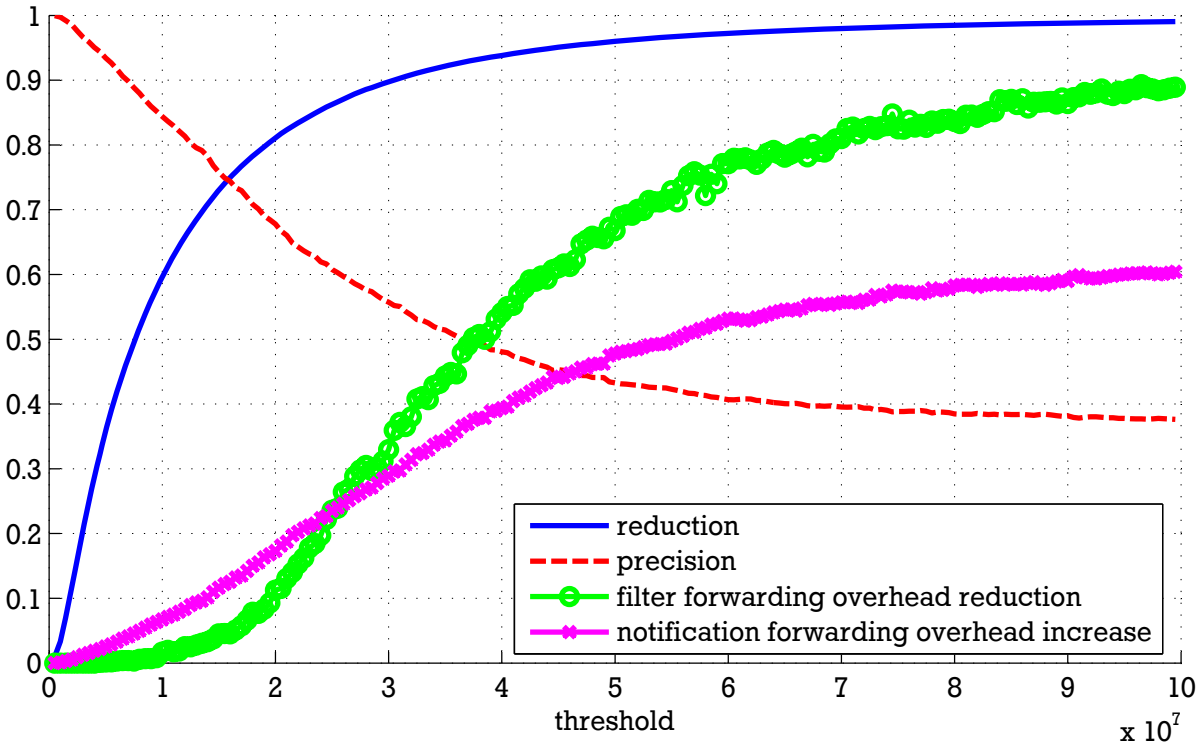
Overhead ratio of a set of 200 random flight subscriptions



Characteristics of a set of 500 random flight subscriptions

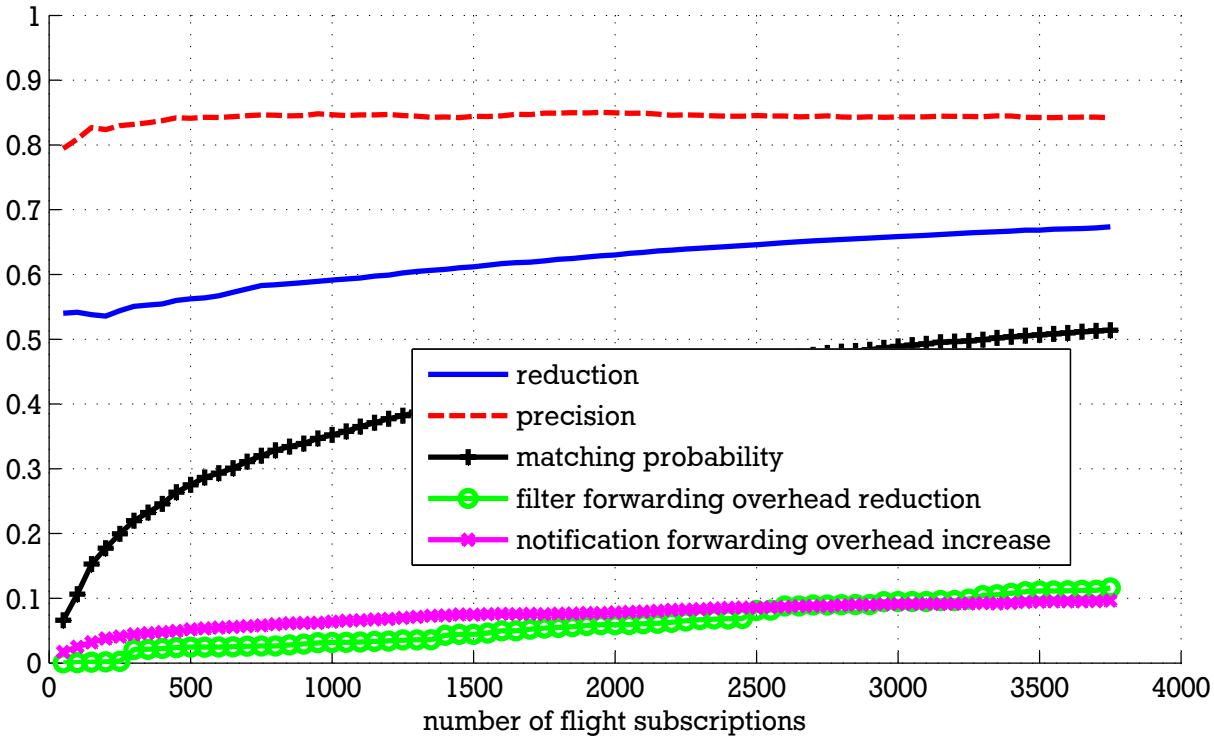


Overhead ratio of a set of 500 random flight subscriptions

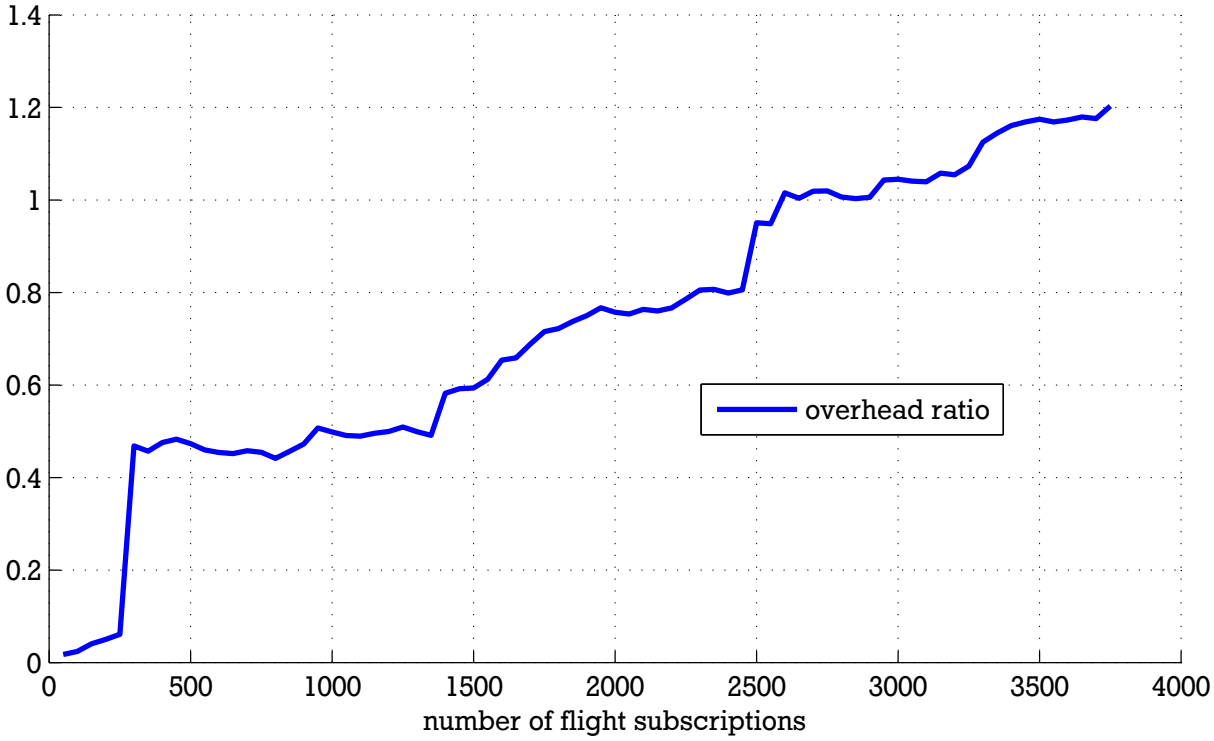


Characteristics of a set of 1000 random flight subscriptions

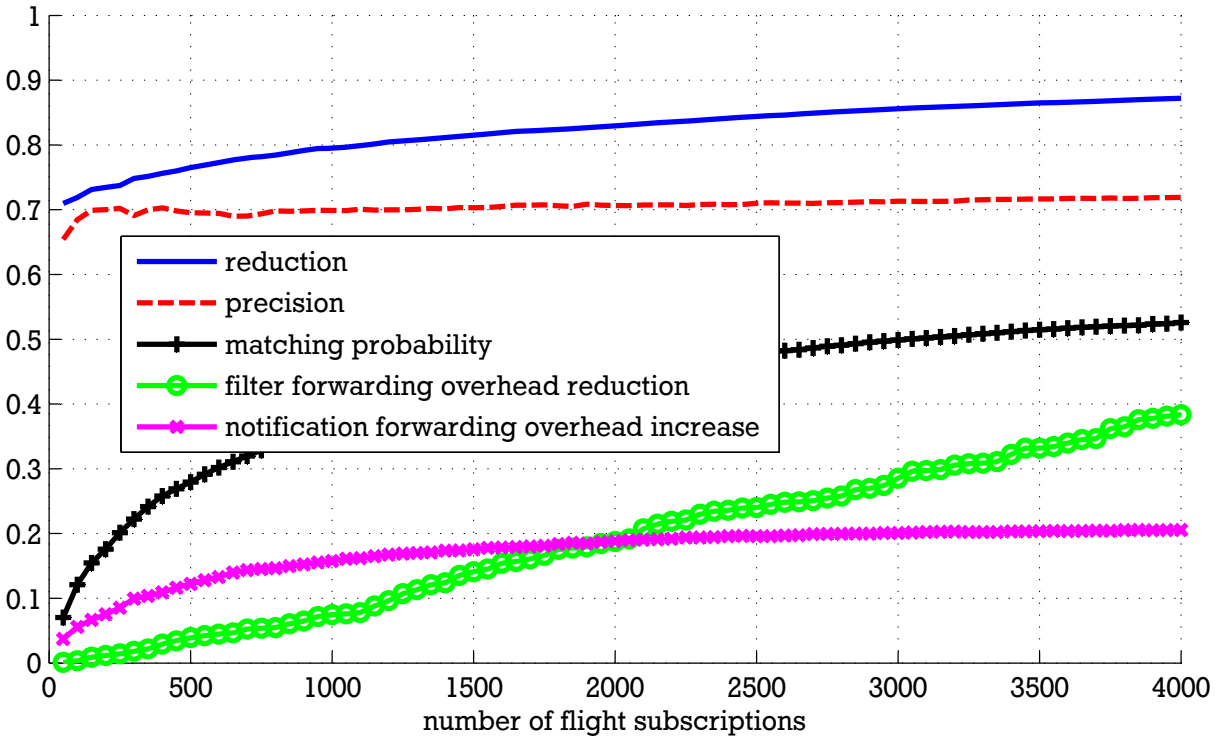
Overhead ratio of a set of 1000 random flight subscriptions



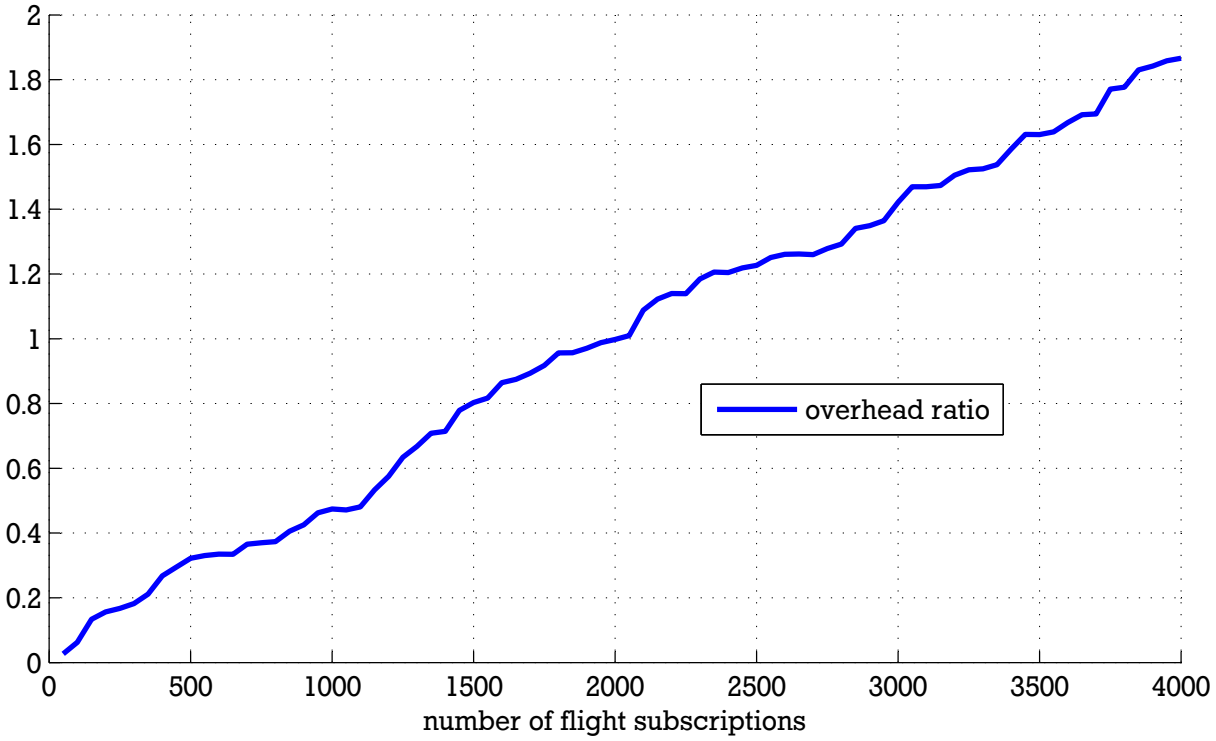
Characteristics of sets of random flight subscriptions at $p_0 = 10^7$



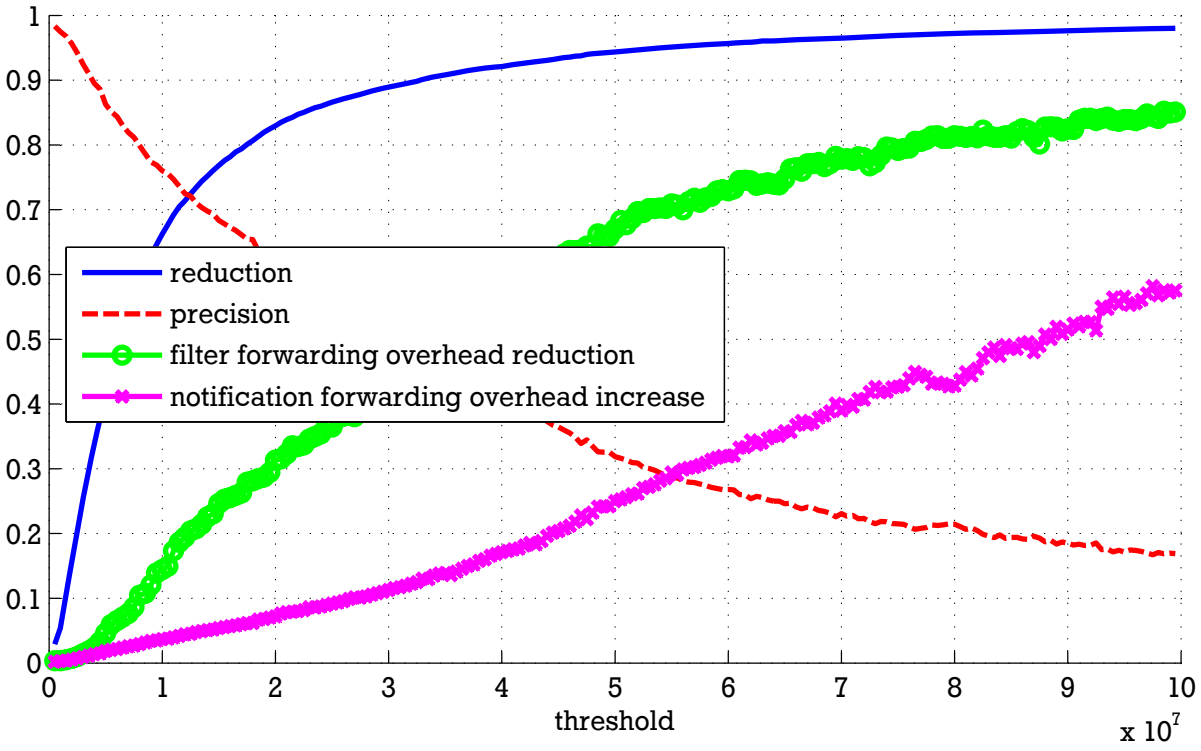
Overhead ratio of sets of random flight subscriptions at $p_0 = 10^7$



Characteristics of sets of random flight subscriptions at $p_0 = 2 \cdot 10^7$

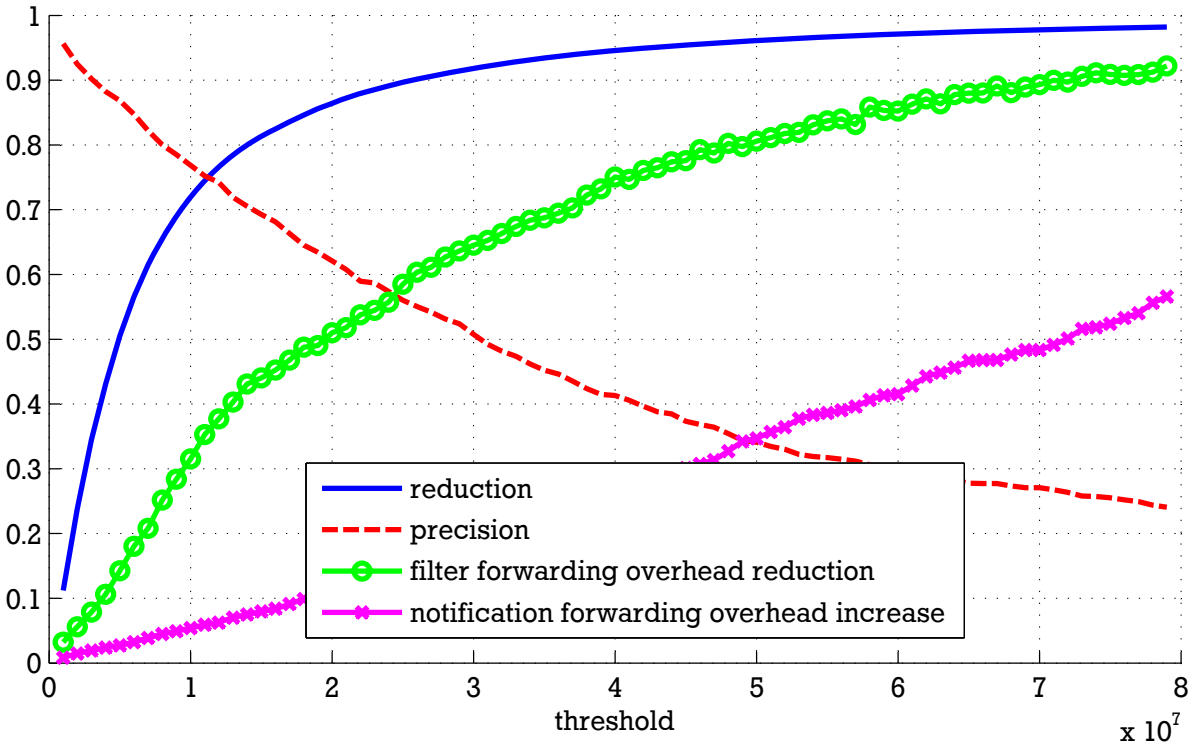


Overhead ratio of sets of random flight subscriptions at $p_0 = 2 \cdot 10^7$

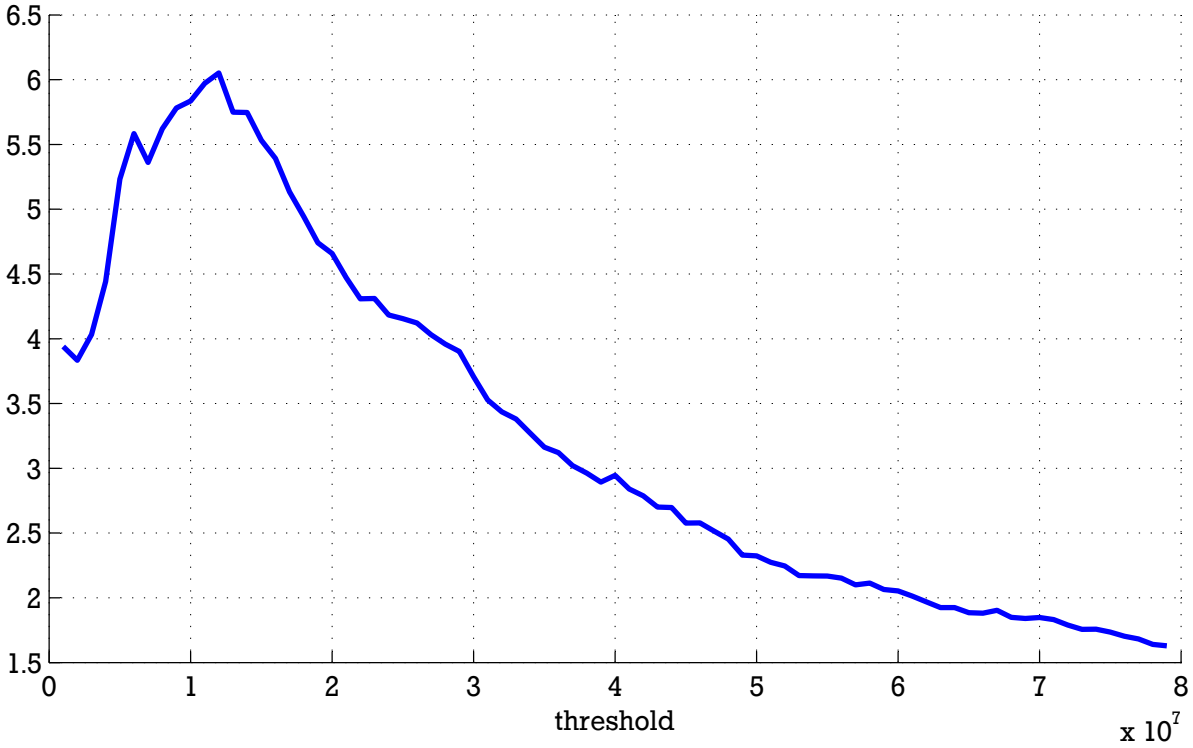


Characteristics of a set of 200 subscriptions for flights from Frankfurt

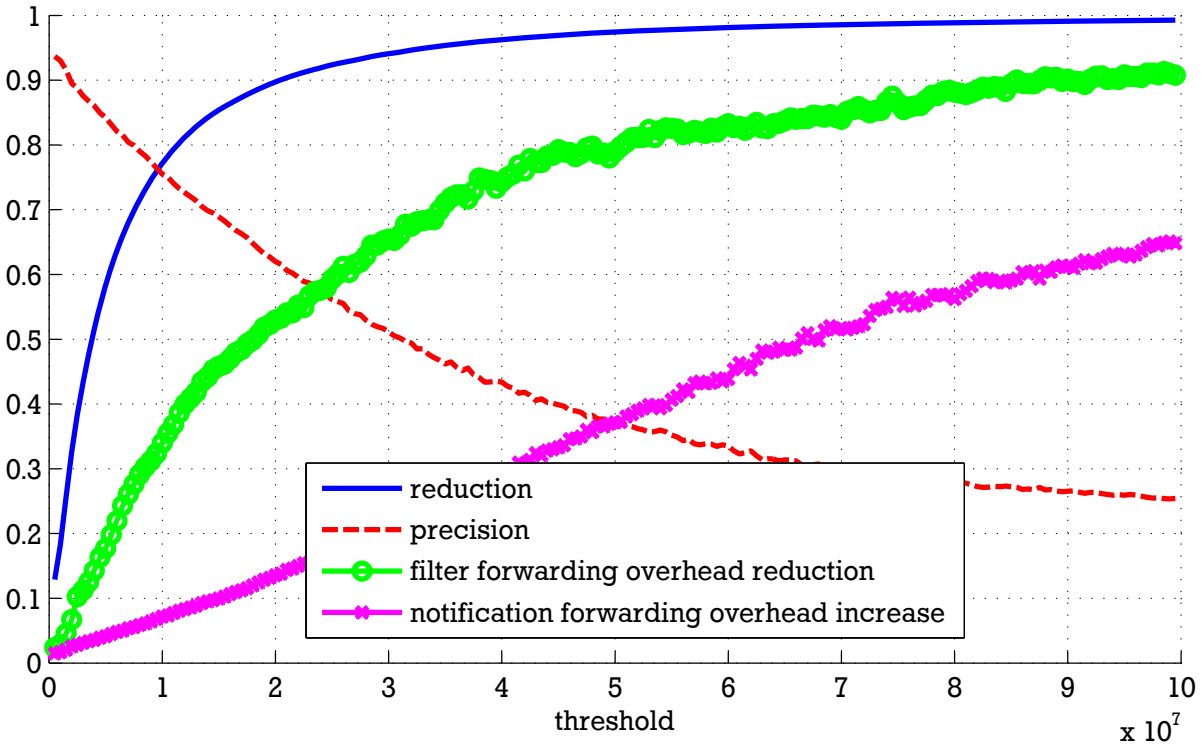
Overhead ratio of a set of 200 subscriptions for flights from Frankfurt



Characteristics of a set of 500 subscriptions for flights from Frankfurt

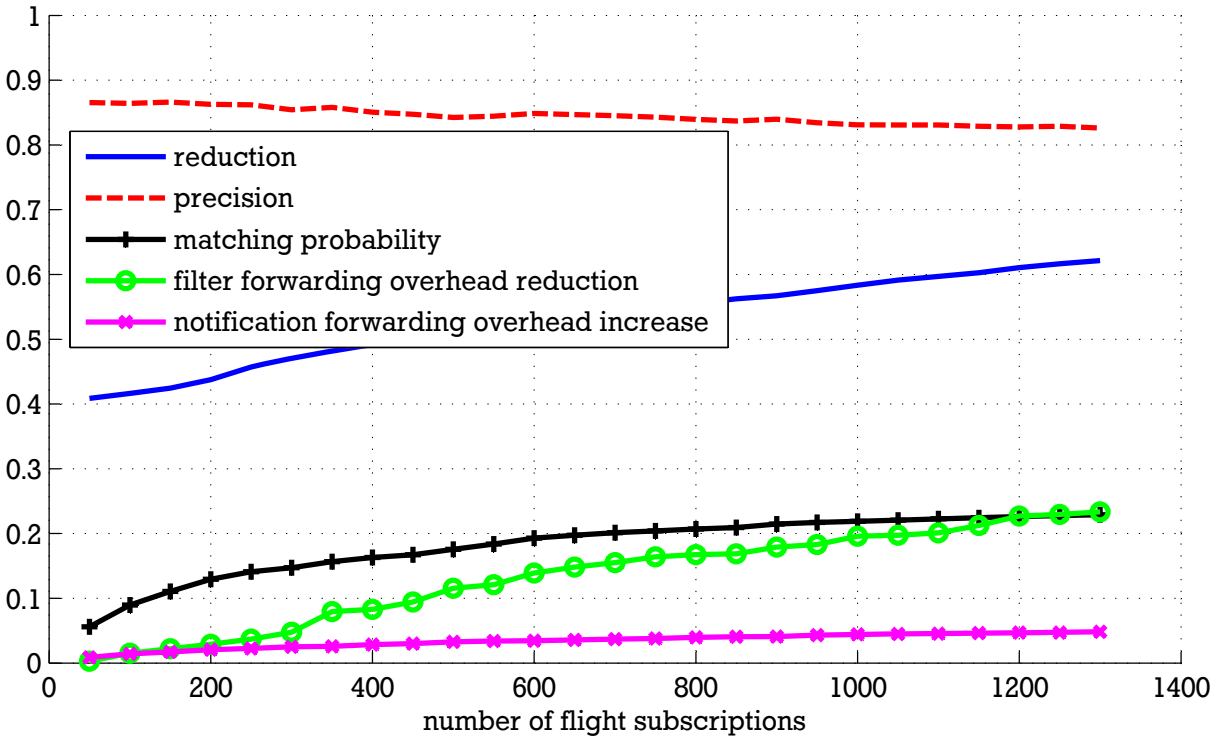


Overhead ratio of a set of 500 subscriptions for flights from Frankfurt

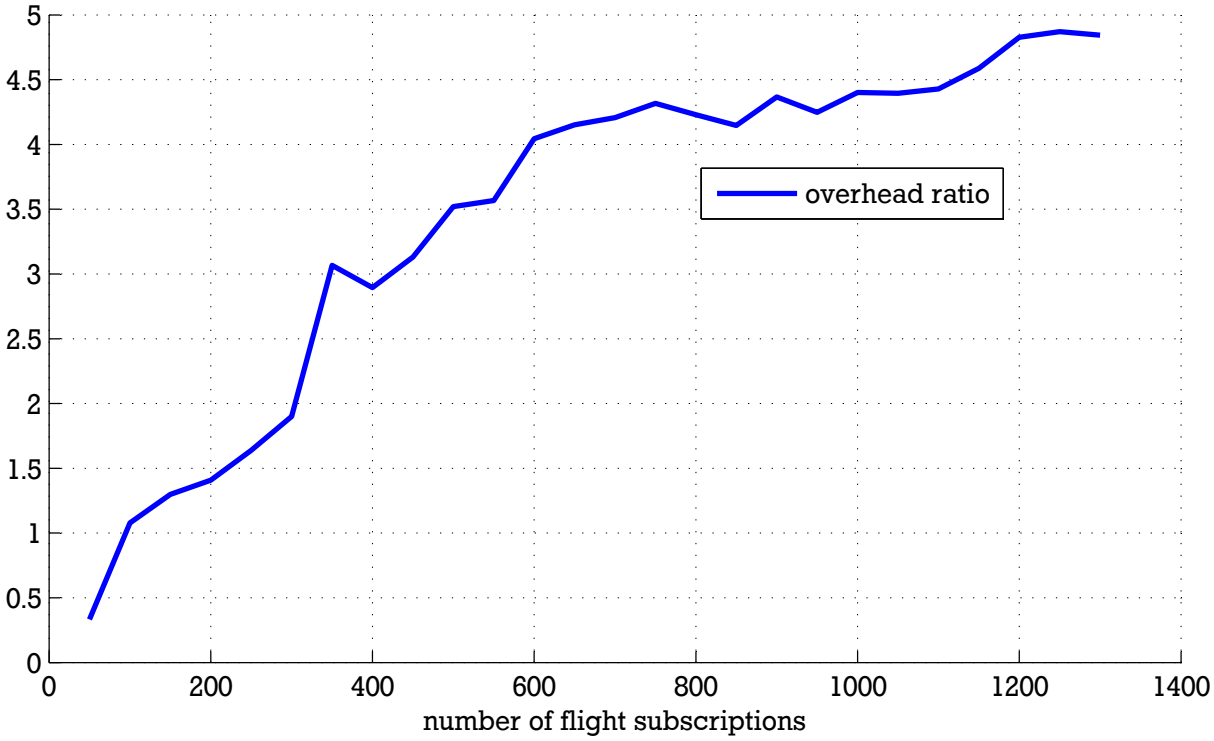


Characteristics of a set of 1000 subscriptions for flights from Frankfurt

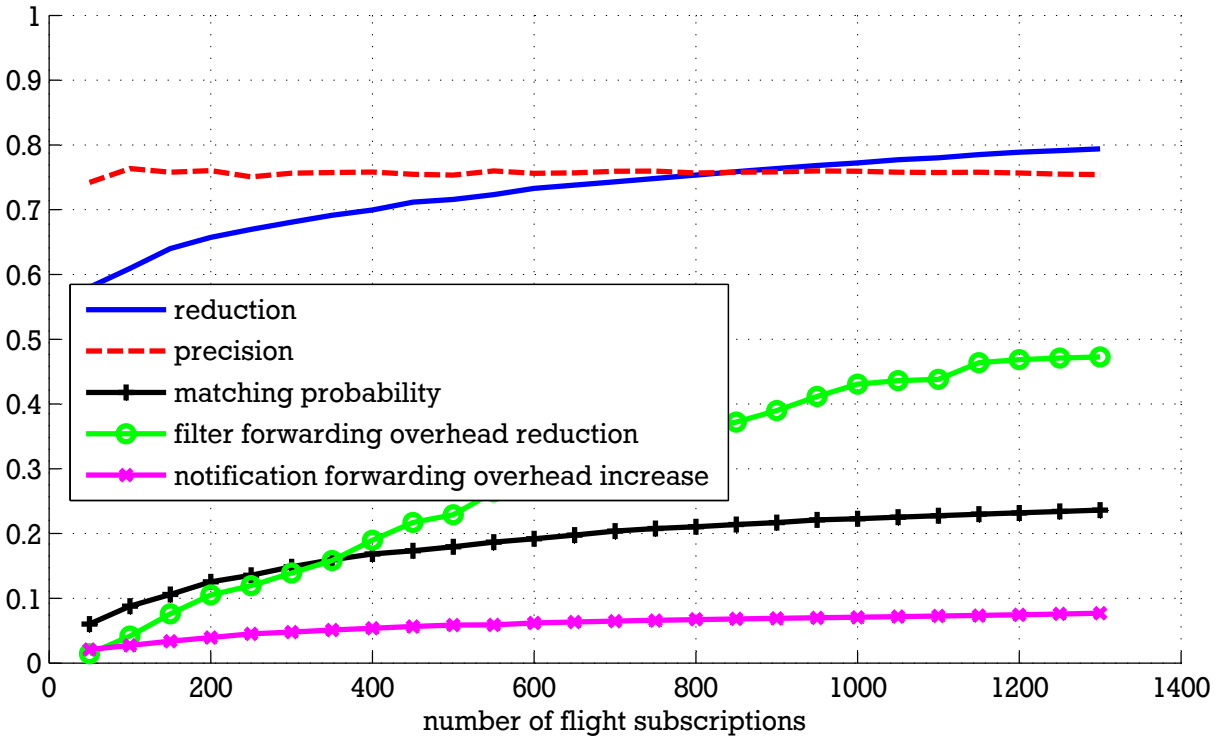
Overhead ratio of a set of 1000 subscriptions for flights from Frankfurt



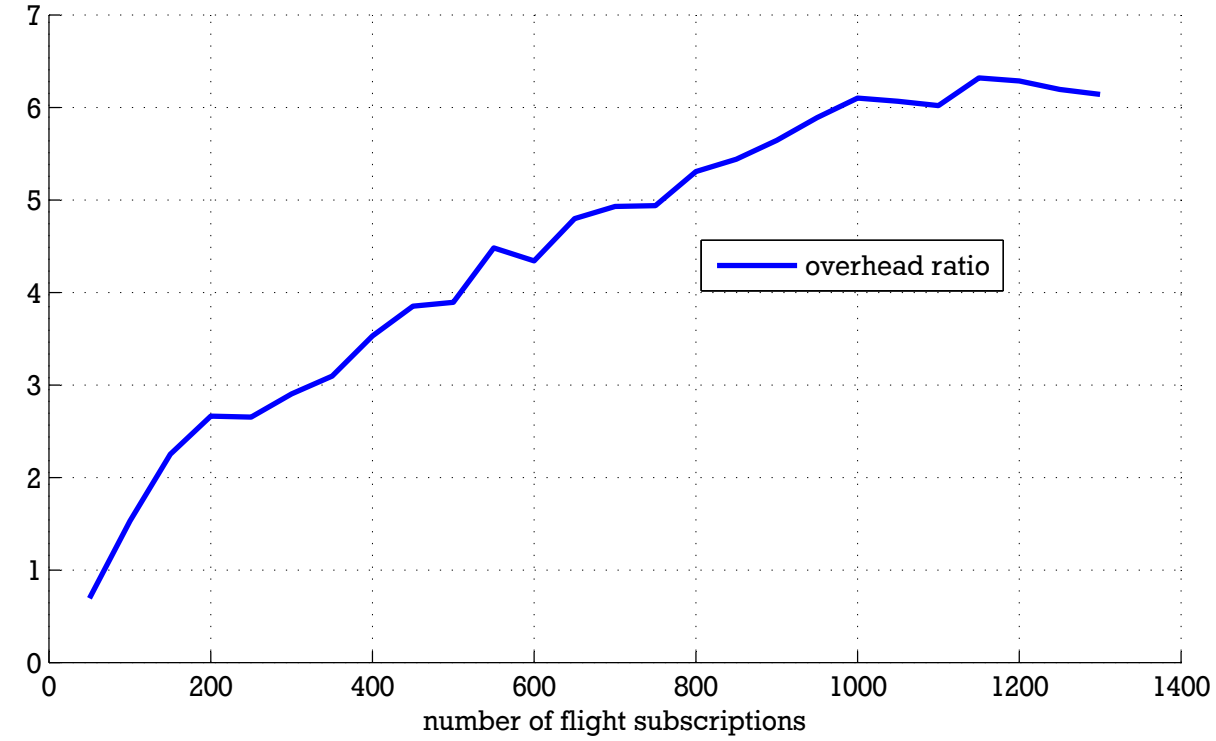
Characteristics of sets of subscriptions for flights from Frankfurt at $p_0 = 5 \cdot 10^6$



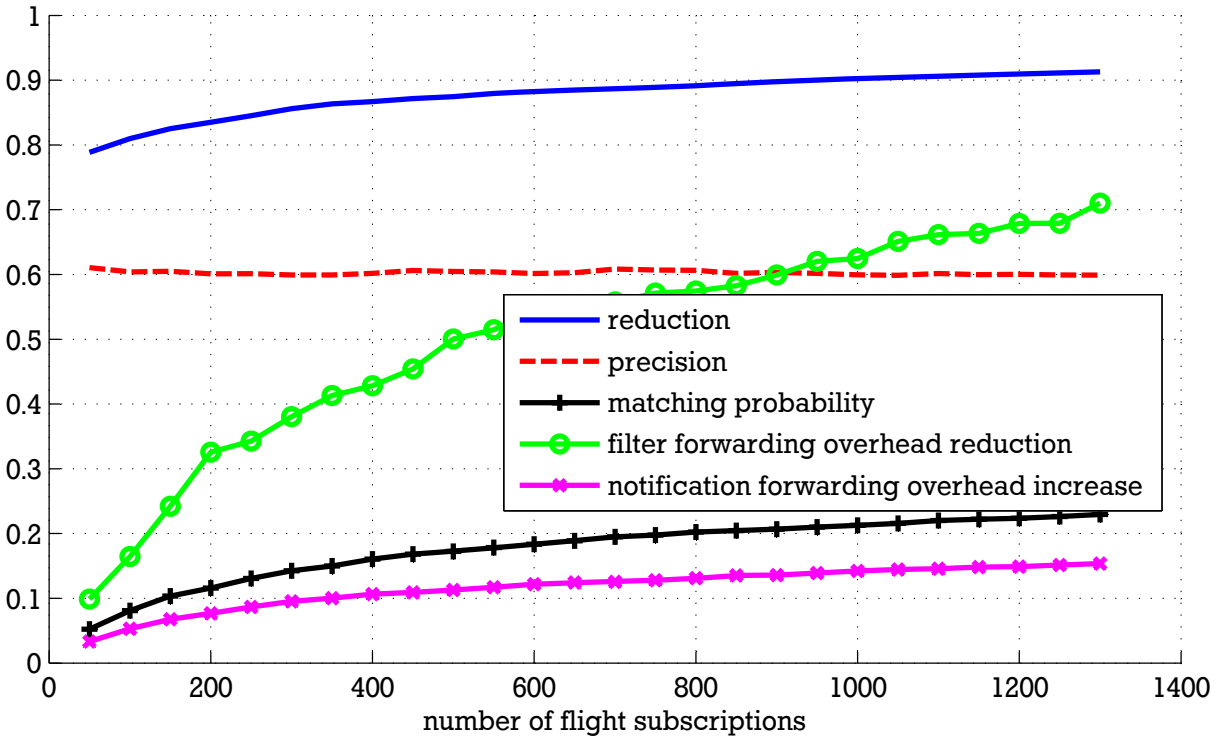
Overhead ratio of sets of subscriptions for flights from Frankfurt at $p_0 = 5 \cdot 10^6$



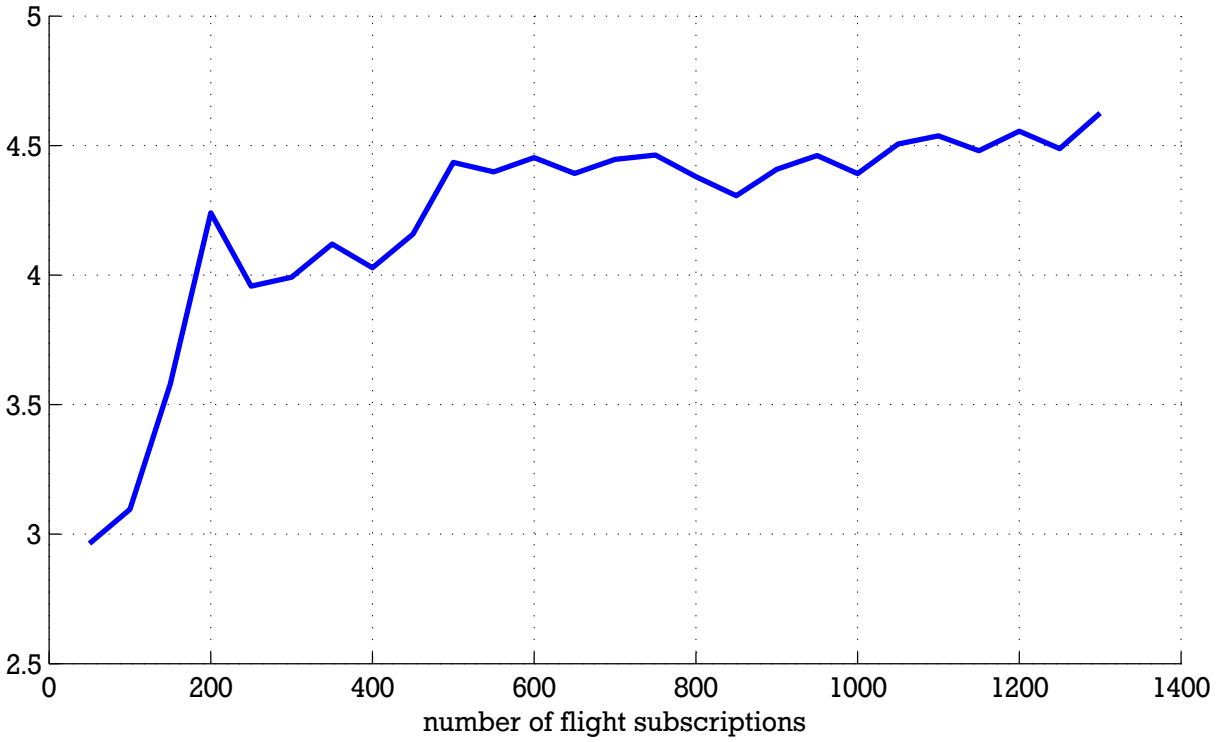
Characteristics of sets of subscriptions for flights from Frankfurt at $p_0 = 10^7$



Overhead ratio of sets of subscriptions for flights from Frankfurt at $p_0 = 10^7$



Characteristics of sets of subscriptions for flights from Frankfurt at $p_0 = 2 \cdot 10^7$



Overhead ratio of sets of subscriptions for flights from Frankfurt at $p_0 = 2 \cdot 10^7$

Table C.2.: Excerpt from experimental results used in effect propagation calculation

type	number	threshold	r	p	m	$O_f = c$	O_n
random	500	$5 \cdot 10^6$	0.3337	0.9241	0.2745	0.0012	0.0221
		10^7	0.5625	0.8411		0.0246	0.0520
		$2 \cdot 10^7$	0.7652	0.6955		0.0394	0.1223
	1500	$5 \cdot 10^6$	0.3687	0.9324	0.4094	0.0082	0.0295
		10^7	0.6119	0.8441		0.0444	0.0748
		$2 \cdot 10^7$	0.8154	0.7031		0.1411	0.1757
similar	500	$5 \cdot 10^6$	0.5149	0.8425	0.1759	0.1155	0.0328
		10^7	0.7159	0.7534		0.2289	0.0588
		$2 \cdot 10^7$	0.8746	0.6046		0.5003	0.1128

Bibliography

[1] Raman Adaikkalavan and Sharma Chakravarthy. SnoopIB: Interval-based event specification and detection for active databases. In *ADBIS: Advances in Databases and Information Systems*, volume 2798 of *Lecture Notes in Computer Science*, pages 190–204, 2003.

[2] Raman Adaikkalavan and Sharma Chakravarthy. Formalization and detection of events using interval-based semantics. In Jayant R. Haritsa and T. M. Vijayaraman, editors, *COMAD*, pages 58–69. Computer Society of India, 2005.

[3] Advisory Council for Aeronautics Research in Europe (ACARE). European aeronautics: A vision for 2020 – meeting society's needs and winning global leadership, January 2001.

[4] Advisory Council for Aeronautics Research in Europe (ACARE). Strategic research agenda. downloadable at `http://www.acare4europe.org`, October 2002. accessed July 9, 2009.

[5] Advisory Council for Aeronautics Research in Europe (ACARE). Strategic research agenda 2. downloadable at `http://www.acare4europe.org`, October 2004. accessed July 9, 2009.

[6] Advisory Council for Aeronautics Research in Europe (ACARE). ACARE 2008 addendum to the strategic research agenda. downloadable at `http://www.acare4europe.org`, 2008. accessed July 9, 2009.

[7] Jordi P. Ahullo, Pedro G. Lopez, and Antonio F. Gomez-Skarmeta. CAPS: Content-based publish/subscribe services for peer-to-peer systems. Fast abstract on DEBS'08, 2008.

[8] Luftfahrthandbuch Deutschland – AIP Germany. Published by AIS Germany operated by Deutsche Flugsicherung DFS, July 2009.

[9] AIXM5 design concepts. `http://www.aixm.aero/public/standard_page/concepts_design.html`. accessed October 30, 2008.

[10] James Alexander. Loxodromes: A rhumb way to go. *Mathematics Magazine*, 77(5):349–356, dec 2004.

[11] James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983.

[12] James F. Allen. Time and time again: the many ways to represent time. *International Journal of Intelligent Systems*, 6:341–355, 1991.

[13] James F. Allen and George Ferguson. Actions and events in interval temporal logic. *J. Log. Comput.*, 4(5):531–579, 1994.

[14] Mehmet Altinel and Michael J. Franklin. Efficient filtering of XML documents for selective dissemination of information. In *VLDB '00: Proceedings of the 26th International Conference on Very Large Data Bases*, pages 53–64, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[15] José Antollini, Mario Antollini, Pablo E. Guerrero, and Mariano Cilia. Extending REBECA to support Concept-Based addressing. In *Proceedings of the Argentinean Symposium on Information Systems (ASIS'04)*, Cordoba, Argentina, September 2004.

[16] Jean Bacon, Ludger Fiege, Rachid Guerraoui, Arno Jacobsen, and Gero Mühl, editors. *1st Intl. Workshop on Distributed Event-Based Systems (DEBS'02)*, Vienna, Austria, 2002. IEEE Press. Published as part of the ICDCS '02 Workshop Proceedings.

[17] Roberto Baldoni, Leonardo Querzoni, and Antonino Virgillito. Distributed event routing in publish/subscribe communication systems: a survey (revised version). Technical report, MIDLAB 1/2006 - Dipartimento di Informatica e Sistemistica "A.Ruberti", Università di Roma la Sapienza, 2006.

[18] M.H. Böhlen, R. Busatto, and C.S. Jensen. Point- versus interval-based temporal data models. *14th International Conference on Data Engineering (ICDE'98)*, 0:192, 1998.

[19] Silvia Bianchi, Pascal Felber, and Maria Gradinariu. Content-based publish/subscribe using distributed R-trees. In Anne-Marie Kermarrec, Luc Bougé, and Thierry Priol, editors, *Proceedings of 13th International Euro-Par Conference (EuroPar 2007)*, volume 4641 of *Lecture Notes in Computer Science*, pages 537–548. Springer, August 2007.

[20] S. Bittner and A. Hinze. Pruning subscriptions in distributed publish/subscribe systems. In *Proceedings of the Twenty-Ninth Australasian Computer Science Conference (ACSC 2006)*, pages 197–206, Hobart, Australia, January 2006. ACS.

[21] S. Bittner and A. Hinze. Subscription tree pruning: A structure-independent routing optimization for general-purpose publish/subscribe systems. Technical Report 01/2006, Computer Science Department, University of Waikato, January 2006.

[22] Sven Bittner. Supporting arbitrary boolean subscriptions in distributed publish/subscribe systems. In *MDS '06: Proceedings of the 3rd international Middleware doctoral symposium*, page 2, New York, NY, USA, 2006. ACM.

[23] Sven Bittner and Annika Hinze. Design and analysis of an efficient distributed event notification service. Technical Report 11/2004, Department of Computer Science, University of Waikato, aug 2004. 2004.

[24] Sven Bittner and Annika Hinze. Dimension-based subscription pruning for publish/subscribe systems. In Annika Hinze and João Pereira, editors, *Proceedings of the 5th International Workshop on Distributed Event-Based Systems (DEBS'06)*, pages 25–25, Lisbon, Portugal, July 2006. IEEE, IEEE.

[25] Brett Brunk. AIXM 5 profile of ISO 19107 spatial schema. downloadable from `http://www.aixm.aero/gallery/content/public/design_review_2/AIXM%205%20Profile%20of%2019107%2020060622.pdf`, June 2006. accessed July 12, 2009.

[26] Brett Brunk and Eduard Porosnicu. Aeronautical Information Exchange Model (AIXM) GIS interoperability through GML. In *ESRI User Conference*, 2005.

[27] Alejandro Buchmann, Christof Bornhövd, Mariano Cilia, Ludger Fiege, Felix Gärtner, Christoph Liebig, Matthias Meixner, and Gero Mühl. *DREAM: Distributed Reliable Event-based Application Management*, pages 319–350. Springer, May 2004.

[28] Antonio Carzaniga. *Architectures for an Event Notification Service Scalable to Wide-area Networks*. PhD thesis, Politecnico di Milano, Milano, Italy, December 1998.

[29] Antonio Carzaniga, Elisabetta Di Nitto, David S. Rosenblum, and Alexander L. Wolf. Issues in supporting event-based architectural styles. In *ISAW '98: Proceedings of the third international workshop on Software architecture*, pages 17–20, New York, NY, USA, 1998. ACM.

[30] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Content-based addressing and routing: A general model and its application. Technical Report CU-CS-902-00, Department of Computer Science, University of Colorado, January 2000.

[31] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3):332–383, August 2001.

[32] Antonio Carzaniga and Alexander L. Wolf. Content-based networking: A new communication infrastructure. In *NSF Workshop on an Infrastructure for Mobile*

and Wireless Systems, number 2538 in Lecture Notes in Computer Science, pages 59–68, Scottsdale, Arizona, October 2001. Springer-Verlag.

[33] Sharma Chakravarthy, V. Krishnaprasad, Eman Anwar, and S.-K. Kim. Composite events for active databases: Semantics, contexts and detection. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *VLDB*, pages 606–617. Morgan Kaufmann, 1994.

[34] Robert G. Chamberlain and William H. Duquette. Some algorithms for polygons on a sphere. Technical Report JPL Publication 07-03, NASA – Jet Propulsion Laboratory, june 2007. Presented at the Association of American Geographers Annual Meeting, San Francisco, California 17Ű21 April 2007.

[35] Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6(5):485–524, 1991.

[36] M. Cilia, M. Antollini, C. Bornhövd, and A. Buchmann. Dealing with heterogeneous data in pub/sub systems: The concept-based approach. In Antonio Carzaniga and Pascal Fenkam, editors, *3rd International Workshop on Distributed Event-Based Systems (DEBS'04)*, Edinburgh, Scotland, UK, May 2004. IEE.

[37] Mariano Cilia, Ludger Fiege, Christian Haul, Andreas Zeidler, and Alejandro Buchmann. Looking into the past: Enhancing mobile publish/subscribe middleware. In H. -Arno Jacobsen, editor, *Proceedings of the 2nd International Workshop on Distributed Event-Based Systems (DEBS'03)*, San Diego, California, June 2003. ACM Press.

[38] Christophe Claramunt and Marius Thériault. Managing time in GIS: An event-oriented approach. In *Proceedings of the International Workshop on Temporal Databases*, pages 23–42, London, UK, 1995. Springer-Verlag.

[39] Eliseo Clementini and Paolino Di Felice. A model for representing topological relationships between complex geometric features in spatial databases. *Inf. Sci.*, 90(1-4):121–136, 1996.

[40] A. G. Cohn and N. M. Gotts. The 'Egg-Yolk' representation of regions with indeterminate boundaries. In P. Burrough and A. M. Frank, editors, *Proceedings, GISDATA Specialist Meeting on Geographical Objects with Undetermined Boundaries*, pages 171–187. Francis Taylor, 1996.

[41] P. Costa and D. Frey. Publish-subscribe tree maintenance over a DHT. In Juergen Dingel and Rob Strom, editors, *4th Intl. Workshop on Distributed Event-Based Systems (DEBS'05)*, pages 414–420, Columbus, Ohio, USA, June 2005. IEEE Press.

[42] Paolo Costa, Matteo Migliavacca, Gian Pietro Picco, and Gianpaolo Cugola. Introducing reliability in content-based publish-subscribe through epidemic algorithms. In Jacobsen [105].

[43] Arturo Crespo, Orkut Buyukkokten, and Hector Garcia-Molina. Query merging: Improving query subscription processing in a multicast environment. *IEEE Trans. on Knowl. and Data Eng.*, 15(1):174–191, 2003.

[44] G. Cugola, E. Di Nitto, and A. Fuggetta. The JEDI event-based infrastructure and its application to the development of the OPSS WFMS. *IEEE Transactions on Software Engineering*, 27(9):827–850, 2001.

[45] Peter H. Dana. Geodetic datum overview. `http://www.colorado.edu/geography/gcraft/notes/datum/datum_f.html`, February 2003. accessed April 25, 2009.

[46] Chris Date and Hugh Darwen. *Temporal Data and the Relational Model*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[47] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, 2nd edition, 2000.

[48] J.E. Dieudonne, H.L. Crane, S.R. Jones, C.J. Smith, S.A. Remillard, and G. Snead. NEO (NextGen 4D TM) provided by SWIM's surveillance SOA (SDN ASP for RNP 4D ops). In *Integrated Communications, Navigation and Surveillance Conference, 2007. ICNS*, pages 1–12, 2007.

[49] Curtis Dyreson, Fabio Grandi, Wolfgang Käfer, Nick Kline, Nikos Lorentzos, Yannis Mitsopoulos, Angelo Montanari, Daniel Nonen, Elisa Peressi, Barbara Pernici, John F. Roddick, Nandlal L. Sarda, Maria Rita Scalas, Arie Segev, Richard Thomas Snodgrass, Mike D. Soo, Abdullah Tansel, Paolo Tiberio, and Gio Wiederhold. A consensus glossary of temporal database concepts. *SIGMOD Rec.*, 23(1):52–64, 1994.

[50] M. J. Egenhofer. A formal definition of binary topological relationships. In *3rd International Conference, FODO 1989 on Foundations of Data Organization and Algorithms*, pages 457–472, New York, NY, USA, 1989. Springer-Verlag New York, Inc.

[51] M. J. Egenhofer and J. R. Herring. A mathematical framework for the definition of topological relationships (extended abstract). In *Proceedings of the 4th International Symposium on Spatial Data Handling (SDH'90)*, pages 803–813, Zurich, July 1990.

[52] Max J. Egenhofer, Eliseo Clementini, and Paolino Di Felice. Topological relations between regions with holes. *International Journal of Geographical Information Systems*, 8(2):129–142, 1994.

[53] Max J. Egenhofer and Robert D. Franzosa. Point-set topological spatial relations. In *International Journal of Geographical Information Systems*, volume 5 of *2*, pages 161–174. Taylor & Francis, 1991.

[54] Max J. Egenhofer and Robert D. Franzosa. On the equivalence of topological relations. *International Journal of Geographical Information Systems*, 9(2):133–152, 1995.

[55] Patrick Eugster. *Type-Based Publish/Subscribe*. PhD thesis, EPFL Lausanne, 2001.

[56] Patrick Eugster. Type-based publish/subscribe: Concepts and experiences. *ACM Trans. Program. Lang. Syst.*, 29(1), 2007.

[57] Patrick Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131, 2003.

[58] Patrick Eugster, Rachid Guerraoui, and Christian Heide Damm. On objects and events. In *OOPSLA*, pages 254–269, 2001.

[59] EUROCAE WG-76 and RTCA SC-206. *Operational Services and Environment Definition (OSED) for Aeronautical Information Services (AIS) and Meteorological (MET) Data Link Services (ED-151)*. EUROCAE, 2007.

[60] EUROCONTROL. *WGS 84 Implementation Manual*. European Organisation for the Safety of Air Navigation, Brussels, Belgium, 2.4 edition, February 1998. `http://www2.icao.int/en/pbn/ICAO%20Documentation/State%20and%20International%20Organization%20Publications/Eurocontrol%20WGS%2084.pdf`, accessed April 22, 2009.

[61] EUROCONTROL. *Aeronautical Information Management Strategy*. European Organisation for the Safety of Air Navigation, Brussels, Belgium, 4 edition, March 2006.

[62] EUROCONTROL. *From AIS to AIM – A Strategic Road Map for Global Change*. European Organisation for the Safety of Air Navigation, Brussels, Belgium, 4 edition, March 2006.

[63] EUROCONTROL. *OPADD: Operating Procedures for AIS Dynamic Data*. European Organisation for the Safety of Air Navigation, Brussels, Belgium, 2.1 edition, March 2007.

[64] EUROCONTROL. Long-term forecast: IFR flight movements 2008-2030. downloadable at http://www.eurocontrol.int/eec/public/standard_page/ETN_2009_1_Growth.html, November 2008. accessed July 9, 2009.

[65] EUROCONTROL. Single european sky ATM research – SESAR in brief. information leaflet published by EUROCONTROL, October 2008.

[66] EUROCONTROL. Challenges of growth 2008 – summary report. downloadable at http://www.eurocontrol.int/eec/public/standard_page/ETN_2009_1_Growth.html, May 2009. accessed July 9, 2009.

[67] EUROCONTROL. *OPADD: Operating Procedures for AIS Dynamic Data*. European Organisation for the Safety of Air Navigation, Brussels, Belgium, 3.0 draft 0.9 edition, February 2009. Proposed Issue.

[68] EUROCONTROL and Federal Aviation Administration. *Aeronautical Information Exchange Model (AIXM) Temporality Model*, 0.5 edition, November 2007.

[69] EUROCONTROL CFMU. 10 million flights in 2007: Analysis and highlights from the CFMU. http://www.eurocontrol.int/corporate/public/news/20080115_figures_CFMU_2007.html. accessed October 31, 2008.

[70] Federal Aviation Administration. *System Wide Information Management (SWIM) Technical Overview*. Federal Aviation Administration, 2.1 edition, March 2008.

[71] Ludger Fiege, Gero Mühl, and Alejandro Buchmann. An architectural framework for electronic commerce applications. In *Informatik 2001: Annual Conference of the German Computer Society*, 2001.

[72] Ludger Fiege, Gero Mühl, and Felix C. Gärtner. Modular event-based systems. *Knowl. Eng. Rev.*, 17(4):359–388, 2002.

[73] GRACE gravity recovery and climate experiment gravity model. http://www.csr.utexas.edu/grace/. accessed June 04, 2009.

[74] Christian Grothe. Updates für Onboard-Datenbanken. Thesis submitted in partial fulfilment of the requirements to be accepted as Doktorand at Fachbereich Informatik, Technische Universität Darmstadt, May 2005.

[75] Christian Grothe. Updating onboard databases. Presentation at AIS Team Technical Subgroup (AISTEC), Brussels, Belgium, November 2005.

[76] Christian Grothe. Onboard database design. Presentation at RTCA SC-206 and EUROCAE WG-44/53 committee meeting, Brussels, Belgium, March 2006. http://www.avmet.com/rtca/brussels.html, accessed April 8, 2009.

[77] Christian Grothe. Study on the applicability of the xNOTAM concept for updating airport mapping database (AMDB) applications – results and findings. Technical report, Technische Universität Darmstadt, 2006. Released under EUROCONTROL contract no. C/1.255/HQ/CG/05.

[78] Christian Grothe. xNOTAM/AMDB study – study on the applicability of the xNOTAM concept for updating airport mapping database (AMDB) applications. Presentation at AIXM5 Public Design Review, Washington D.C., USA, February 2006.

[79] Christian Grothe. AIXM5 procedure model verification and gap analysis report. Technical report, Technische Universität Darmstadt, 2007. Released under EUROCONTROL contract no. PROC/NG/C06/11261.

[80] Christian Grothe. AIXM5 terminal procedure model study – verification of the AIXM5 procedures model through cockpit simulation applications. Presentation at AIXM Users Conference, Washington D.C., USA, March 2007.

[81] Christian Grothe. Amdb/xnotam study – study on the applicability of the xNOTAM concept for updating airport mapping database (AMDB) applications. Presentation at AIXM Users Conference, Washington D.C., USA, March 2007.

[82] Christian Grothe. Deriving a relational DB model from the AIXM CM. Presentation at AIXM Training Workshop held in conjunction with AIXM Users Conference, Washington D.C., USA, March 2007.

[83] Christian Grothe. AIXM5 – integrating time into the core of the aeronautical geoweb. In *GeoWeb 2008*, Vancouver, Canada, July 2008.

[84] Christian Grothe. GML 3.2.1 CR for CurveInterpolationType. `https://portal.opengeospatial.org/files/?artifact_id=31244`, November 2008. Document 08-194, Change Request for GML addressed to the Open Geospatial Consortium.

[85] Christian Grothe, Siegfried Bauer, and Uwe Klingauf. Overcoming the media breaks in AIS: Dissemination of operational change notifications by xNOTAM. In Otto Estorff, editor, *International Workshop on Aircraft System Technologies AST 2007*, pages 303–312. Shaker Verlag, March 2007.

[86] Christian Grothe, Andre Nathaus, Tsvetan Penev, and Uwe Klingauf. AIXM5 als Treiber des Paradigmenwechsels in den aeronautischen Informationsdiensten. In *Deutscher Luft- und Raumfahrtkongress DLRK 2008*, September 2008.

[87] Christian Grothe and Jochen Schaab. An evaluation of kernel density estimation and support vector machines for automated generation of footprints for imprecise regions from geotags. In *PLACE'08: International Workshop on Computational Models of Place*, September 2008.

[88] Christian Grothe and Jochen Schaab. Automated footprint generation from geotags with kernel density estimation and support vector machines. *Spatial Cognition and Computation – Special Issue: Computational Models of Place*, 9(3):195–211, 2009.

[89] Pablo Guerrero. Looking into the past: Enhancing mobile publish/subscribe middleware. Master's thesis, Faculty of Sciences, UNICEN, Tandil, Argentina, October 2004. In collaboration with the Databases and Distributed Systems Group, Technische Universität Darmstadt.

[90] Eric Haines. Point in polygon strategies. In Paul Heckbert, editor, *Graphics Gems IV*, pages 24–46. Academic Press Professional, Inc., San Diego, CA, USA, 1994.

[91] Sidath Handurukande, Patrick Eugster, Pascal Felber, and Rachid Guerraoui. Event systems: How to have ones cake and eat it too. In Bacon et al. [16]. Published as part of the ICDCS '02 Workshop Proceedings.

[92] Manfred Hauswirth and Mehdi Jazayeri. A component and communication model for push systems. *SIGSOFT Softw. Eng. Notes*, 24(6):20–38, 1999.

[93] James Higginbotham, Fabio Pianesi, and Achille C. Varzi, editors. *Speaking of Events*. Oxford University Press, New York, NY, USA, 2000.

[94] Linda L. Hill. *Georeferencing: The Geographic Associations of Information*. The MIT Press, 2006.

[95] Bernhard Hofmann-Wellenhof, Herbert Lichtenegger, and Elmar Wasle. *GNSS - Global Navigation Satellite Systems: GPS, GLONASS, Galileo & more*. Springer, 2007.

[96] ICAO. *World Geodetic System 1984 (WGS-84) Manual — Doc 9674*. International Civil Aviation Organization, Montréal, Canada, 2nd edition, 2002.

[97] ICAO. *Aeronautical Information Services Manual — Doc 8126*. International Civil Aviation Organization, Montréal, Canada, 6th edition, 2003.

[98] ICAO. *Aeronautical Information Services — Annex 15 to the Convention on International Civil Aviation*. International Civil Aviation Organization, Montréal, Canada, 12th edition, July 2004.

[99] ICAO. Protocol of the first meeting of the aeronautical information services-aeronautical information management study group (ais-aimsg). http://www.icao.int/anb/AIM/SG/meetings/2008/AISAIMSG1/SoD/AIS-AIMSG.1.SoD.en.pdf, December 2008. accessed January 15, 2009.

[100] ISO. *ISO 19101:2002: Geographic information – Reference model*. International Organization for Standardization, Geneva, Switzerland, 2002.

[101] ISO. *ISO 19108:2002: Geographic information – Temporal schema*. International Organization for Standardization, Geneva, Switzerland, 2002.

[102] ISO. *ISO 19107:2003: Geographic information – Spatial schema*. International Organization for Standardization, Geneva, Switzerland, 2003.

[103] ISO. *ISO 19111:2003: Geographic information – Spatial referencing by coordinates*. International Organization for Standardization, 2003.

[104] ISO. *ISO 19136:2007: Geographic information – Geography Markup Language (GML)*. International Organization for Standardization, Geneva, Switzerland, 2007.

[105] Arno Jacobsen, editor. *2nd Intl. Workshop on Distributed Event-Based Systems (DEBS '03)*, San Diego, CA, USA, June 2003. ACM Press.

[106] Michael A. Jaeger. *Self-Managing Publish/Subscribe Systems*. PhD thesis, Berlin Institute of Technology, Berlin, Germany, December 2007.

[107] Michael A. Jaeger, Helge Parzyjegla, Gero Mühl, and Klaus Herrmann. Self-organizing broker topologies for publish/subscribe systems. In *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, pages 543–550, New York, NY, USA, 2007. ACM.

[108] Sanjay Dominik Jena and Jackson Roehrig. A java implementation of the opengis™feature geometry abstract specification (iso 19107 spatial schema). Technical report, University of Applied Sciences Cologne, Institute for Technology in the Tropics, July 2007.

[109] Christian S. Jensen, James Clifford, Shashi K. Gadia, Arie Segev, and Richard T. Snodgrass. A glossary of temporal database concepts. *SIGMOD Record*, 21(3):35–43, 1992.

[110] Christopher B. Jones, Ross S. Purves, Paul D. Cloughs, and Hideo Joho. Modelling vague places with knowledge from the web. *International Journal of Geographical Information Science*, 22(10):1045–1065, 2008.

[111] JPDO. Next generation air transportation system – in brief. information leaflet published by the Joint Planning and Development Office (JPDO), 2006.

[112] JPDO. Concept of operations for the next generation air transportation system. published by the Joint Planning and Development Office (JPDO), 2007.

[113] Ibrahim Kamel and Christos Faloutsos. Hilbert R-tree: An improved R-tree using fractals. In *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, pages 500–509, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[114] Alfons Kemper and Mechtild Wallrath. An analysis of geometric modeling in database systems. *ACM Comput. Surv.*, 19(1):47–91, 1987.

[115] Anne-Marie Kermarrec and Peter Triantafillou. Large-scale publish/subscribe systems: State of the art and research directions. Tutorial given at the Second international conference on Distributed event-based systems (DEBS'08), 2008. `http://event-based.org/content/debs08_tutorials/kermarrec_triantafillou-large_scale_publish_subscribe_systems.pdf`, accessed September 15, 2009.

[116] Eugene F. Krause. *Taxicab Geometry*. Dover Publications, 1987.

[117] Patrick Ky and Bernard Miaillier. SESAR: towards the new generation of air traffic management systems in europe. *ATCA Journal of Air Traffic Control*, 1, 2006.

[118] Ron Lake, David Burggraf, Milan Trninic, and Laurie Rae. *GML – Geography MarkUp Language: Foundation for the Geo-Web*. John Wiley & Sons, Hoboken, NJ, USA, April 2004.

[119] Gail Langran. States, events and evidence: The principle entities of a temporal gis. In *Proc. of GIS/LIS '92*, San Jose, California, 1992.

[120] F. G. Lemoine, S. C. Kenyon, J. K. Factor, R.G. Trimmer, N. K. Pavlis, D. S. Chinn, C. M. Cox, S. M. Klosko, S. B. Luthcke, M. H. Torrence, Y. M. Wang, R. G. Williamson, E. C. Pavlis, R. H. Rapp, and T. R. Olson. The development of the joint NASA GSFC and NIMA geopotential model EGM96. Technical Report TP-1998-206861, NASA Goddard Space Flight Center, Greenbelt, Maryland, 20771 USA, July 1998.

[121] Ulf Leonhardt. *Supporting Location-Awareness in Open Distributed Systems*. PhD thesis, University of London, 1998.

[122] Guoli Li, Alex Cheung, Shunag Hou, Songlin Hu, Vinod Muthusamy, Reza Sherafat, Alex Wun, Hans-Arno Jacobsen, and Serge Manovski. Historic data access in publish/subscribe. In Hans-Arno Jacobsen, Gero Mühl, and Michael A. Jaeger, editors, *Proceedings of the Inaugural Conference on Distributed Event-Based Systems*, pages 80–84, New York, NY, USA, June 2007. ACM Press. Demo paper.

[123] Guoli Li, Shuang Hou, and Hans-Arno Jacobsen. A unified approach to routing, covering and merging in publish/subscribe systems based on modified binary decision diagrams. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pages 447–457. IEEE, 2005.

[124] Christoph Liebig, Mariano Cilia, and Alejandro Buchmann. Event composition in time-dependent distributed systems. In *Proceedings of the 4th Intl. Conference on Cooperative Information Systems (CoopIŚ99)*, pages 70–78. IEEE Computer Society, September 1999.

[125] Nikos A. Lorentzos. The interval extended relational model and its application to valid time databases. In Abdullah Uz Tansel, James Clifford, Shashi Gadia, Sushil Jajodia, Arie Segev, and Richard Snodgrass, editors, *Temporal Databases: Theory, Design, and Implementation*, pages 67–91. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1993.

[126] Nikos A. Lorentzos and V. J. Kollias. The handling of depth and time intervals in soil-information systems. *Comput. Geosci.*, 15(3):395–401, 1989.

[127] Nikos A. Lorentzos and Yannis G. Mitsopoulos. SQL extension for interval data. *IEEE Trans. on Knowl. and Data Eng.*, 9(3):480–499, 1997.

[128] Gary Luckenbaugh, Scott Landriau, Jon Dehn, and Sid Rudolph. Service oriented architecture for the next generation air transportation system. In *Integrated Communications, Navigation and Surveillance Conference ICNS '07*, 2007.

[129] V. Lum, P. Dadam, R. Erbe, J. Guenauer, P. Pistor, G. Walch, H. Werner, and J. Woodfill. Designing dbms support for the temporal dimension. In *SIGMOD '84: Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, pages 115–130, New York, NY, USA, 1984. ACM Press.

[130] Jere S. Meserole and John W. Moore. What is system wide information management (SWIM)? In *25th Digital Avionics Systems Conference, 2006 IEEE/AIAA*, pages 1–8, 2006.

[131] Gero Mühl, Ludger Fiege, and Alejandro P. Buchmann. Evaluation of cooperation models for electronic business. In *Information Systems for E-Commerce, Confer-*

ence of German Society for Computer Science / EMISA, pages 81–94, November 2000. ISBN 3-85487-194-5.

[132] David M. Mount. Geometric intersection. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 857–876. Chapman & Hall/CRC, Boca Raton, FL, USA, 2nd edition, 2004.

[133] Gero Mühl. Generic constraints for content-based publish/subscribe systems. In C. Batini, F. Giunchiglia, P. Giorgini, and M. Mecella, editors, *Proceedings of the 6th International Conference on Cooperative Information Systems (CoopIS'01)*, volume 2172 of *LNCS*, pages 211–225, Trento, Italy, September 2001. Springer-Verlag.

[134] Gero Mühl. *Large-Scale Content-Based Publish/Subscribe Systems*. PhD thesis, Technische Universität Darmstadt, September 2002.

[135] Gero Mühl and Ludger Fiege. Supporting covering and merging in Content-Based publish/subscribe systems: Beyond name/value pairs. *IEEE Distributed Systems Online (DSOnline)*, 2(7), 2001.

[136] Gero Mühl, Ludger Fiege, and Alejandro P. Buchmann. Filter similarities in Content-Based publish/subscribe systems. In H. Schmeck, T. Ungerer, and L. Wolf, editors, *International Conference on Architecture of Computing Systems (ARCS)*, volume 2299 of *LNCS*, pages 224–238, Karlsruhe, Germany, April 2002. Springer-Verlag.

[137] Gero Mühl, Ludger Fiege, Felix C. Gärtner, and Alejandro P. Buchmann. Evaluating advanced routing algorithms for Content-Based publish/subscribe systems. In A. Boukerche, S. K. Das, and S. Majumdar, editors, *The 10th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2002)*, pages 167–176, Fort Worth, TX, USA, October 2002. IEEE Press.

[138] Gero Mühl, Ludger Fiege, and Peter R. Pietzuch. *Distributed Event-Based Systems*. Springer, August 2006.

[139] NIMA. Department of defense world geodetic system 1984: Its definition and relationships with local geodetic systems, 3rd edition. Technical Report TR8350.2, National Imagery and Mapping Agency, 1997.

[140] Object Management Group (OMG). Event service specification, version 1.2. Document formal/04-10-02. Downloadable from `http://www.omg.org/technology/documents/formal/event_service.htm`, October 2004. accessed September 11, 2009.

[141] Brian Oki, Manfred Pfluegl, Alex Siegel, and Dale Skeen. The information bus: an architecture for extensible distributed systems. In *SOSP '93: Proceedings of the fourteenth ACM symposium on Operating systems principles*, pages 58–68, New York, NY, USA, 1993. ACM.

[142] Open Geospatial Consortium Inc. *OpenGIS Implementation Specification for Geographic information – Simple feature access – Part 1:Common architecture*. OGC, 1.1.0 edition, November 2005. `http://www.opengeospatial.org/standards/sfa`, accessed April 28, 2009.

[143] Open Geospatial Consortium Inc. *Geography Markup Language (GML) simple features profile*. OGC, 1.0 edition, April 2006. `http://www.ogcnetwork.net/gml-sf`, accessed April 28, 2009.

[144] Open Geospatial Consortium Inc. *OpenGIS Geography Markup Language (GML) Encoding Standard*. OGC, 3.2.1 edition, August 2007. `http://www.opengeospatial.org/standards/gml`, accessed April 28, 2009.

[145] Joseph O'Rourke. *Computational geometry in C (2nd ed.)*. Cambridge University Press, New York, NY, USA, 1998.

[146] Tsvetan Penev. Supporting geo-based routing in pub/sub middleware. Master's thesis, Technische Universität Darmstadt, August 2007.

[147] Alexandre Petrovsky. AIRAC adherence. available online via `http://www.eurocontrol.int/aim/public/standard_page/qm_airacadh_intro.html`. accessed August 21, 2008.

[148] Fabio Pianesi and Achille C. Varzi. Events and event talk. In Higginbotham et al. [93], pages 3–47.

[149] Peter Pietzuch and Jean Bacon. Hermes: A distributed event-based middleware architecture. In Bacon et al. [16]. Published as part of the ICDCS '02 Workshop Proceedings.

[150] Franco P. Preparata and Michael I. Shamos. *Computational Geometry: An Introduction*. Springer, August 1985.

[151] S. Puechmorel and D. Delahaye. 4d trajectories: A functional data perspective. In *Digital Avionics Systems Conference 2007*, volume 26, pages 1.C.6–1–1.C.6–12, 2007.

[152] Costin Raiciu, David S. Rosenblum, and Mark Handley. Revisiting content-based publish/subscribe. In *ICDCSW '06: Proceedings of the 26th IEEE International*

ConferenceWorkshops on Distributed Computing Systems, page 19, Washington, DC, USA, 2006. IEEE Computer Society.

[153] Markus Schneider. Spatial data types – a survey. In *Spatial Data Types for Database Systems*, volume 1288 of *Lecture Notes in Computer Science*, chapter 2, pages 11–83. Springer, 1997.

[154] Steven Schockaert, Martine De Cock, and Etienne E. Kerre. Automatic acquisition of fuzzy footprints. In *Proceedings of the OTM 2005 Workshops, On the Move to Meaningful Internet Systems 2005*, pages 1077–1086, October – November 2005.

[155] W. Segall and D. Arnold. Elvin has left the building: A publish/subscribe notification service with quenching. In *Proceedings of the 1997 Australian UNIX Users Group, Brisbane, Australia, September 1997.*, 1997. http://elvin.dstc.edu.au/doc/papers/auug97/AUUG97.html.

[156] SESAR Consortium. Air transport framework – the current situation (milestone deliverable 1). Technical report, SESAR Consortium, July 2006.

[157] SESAR Consortium. The ATM masterplan (milestone deliverable 5). Technical Report DLM-0612-001-02-00a, SESAR Consortium, 2007.

[158] SESAR Consortium. The ATM target concept (milestone deliverable 3). Technical Report DLM-0612-001-02-00a, SESAR Consortium, 2007.

[159] Richard Snodgrass and Ilsoo Ahn. A taxonomy of time in databases. In Shamkant B. Navathe, editor, *Proceedings of the 1985 ACM SIGMOD International Conference on Management of Data*, pages 236–246, New York, NY, USA, 1985. ACM Press.

[160] Richard Snodgrass and Ilsoo Ahn. Temporal databases. *Computer*, 19(9):35–41, 1986.

[161] John Snyder. *Map Projections–A Working Manual*. United States Government Printing, February 1983.

[162] Ivan E. Sutherland, Robert F. Sproull, and Robert A. Schumacker. A characterization of ten hidden-surface algorithms. *ACM Comput. Surv.*, 6(1):1–55, 1974.

[163] Sasu Tarkoma and Jaakko Kangasharju. Filter merging for efficient information dissemination. In Robert Meersman, Zahir Tari, Mohand-Said Hacid, John Mylopoulos, Barbara Pernici, Özalp Babaoglu, Hans-Arno Jacobsen, Joseph P. Loyall, Michael Kifer, and Stefano Spaccapietra, editors, *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, volume 3760 of *Lecture Notes in Computer Science*, pages 274–291. Springer, 2005.

[164] M.S. Taylor. System-wide information management for aeronautical communications. In *Digital Avionics Systems Conference DASC '04*, volume 2, 2004.

[165] Wesley W. Terpstra, Stefan Behnel, Ludger Fiege, Andreas Zeidler, and Alejandro Buchmann. A peer-to-peer approach to content-based publish/subscribe. In Jacobsen [105].

[166] J. Teutsch and E. Hoffman. Aircraft in the future atm system - exploiting the 4d aircraft trajectory. In *Digital Avionics Systems Conference DASC 04*, pages 3.B.2–31–3.B.2–22, 2004.

[167] Sam Van der Stricht. D-AIM: Initial validation of the AIM concept. Presentation at Global AIM Congress, Singapore, downloadable from `http://www.eurocontrol.int/aim/public/standard_page/globalaim2008_presentations_workshop.html`, June 2008.

[168] Vivid Solutions. *JTS Topology Suite Technical Specifications*, 1.4 edition, 2003.

[169] Nigel Walford. *Geographical Data*. Crystal Dreams Pub, 2002.

[170] Yi-Min Wang, Lili Qiu, Dimitris Achlioptas, Gautam Das, Paul Larson, and Helen J. Wang. Subscription partitioning and routing in content-based publish/subscribe networks. In *16th International Symposium on DIStributed Computing (DISC'02)*, Toulouse, France, October 2002.

[171] Yi-Min Wang, Lili Qiu, Chad Verbowski, Dimitris Achlioptas, Gautam Das, and Paul Larson. Summary-based routing for content-based event distribution networks. *SIGCOMM Comput. Commun. Rev.*, 34(5):59–74, 2004.

[172] K.D. Wichman, J.K. Klooster, O.F. Bleeker, and R.M. Rademaker. Flight validation of downlinked flight management system 4d trajectory. In *Digital Avionics Systems Conference DASC '07*, pages 1.D.1–1–1.D.1–10, 2007.

[173] Wikipedia. Geographic coordinate system. `http://en.wikipedia.org/wiki/Geographic_coordinate_system`. accessed August 27, 2008.

[174] Wikipedia. Geoid. `http://en.wikipedia.org/wiki/Geoid`. accessed September 29, 2009.

[175] Wikipedia. Jordan curve theorem. `http://en.wikipedia.org/wiki/Jordan_curve_theorem`. accessed September 02, 2008.

[176] Wikipedia. Theorema egregium. `http://en.wikipedia.org/wiki/Theorema_Egregium`. accessed August 28, 2008.

[177] Wikipedia. Traveling salesman problem. http://en.wikipedia.org/wiki/Traveling_salesman_problem. accessed November 05, 2009.

[178] Ed Williams. Aviation formulary v1.43. http://williams.best.vwh.net/avform.htm. accessed August 26, 2008.

[179] Eiko Yoneki and Jean Bacon. An adaptive approach to content-based subscription in mobile ad hoc networks. In *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications, Workshop on Mobile Peer-to-Peer Computing (MP2P'04)*, pages 92–97. IEEE Computer Society, March 2004.

[180] Andreas Zeidler. *A Distributed Publish/Subscribe Notification Service for Pervasive Environments*. PhD thesis, Technische Universität Darmstadt, 2004. http://elib.tu-darmstadt.de/diss/000519.

Lebenslauf des Autors

Name	Christian Grothe	
Geburtsdatum	22.11.1977	
Geburtsort	Hadamar	
Nationalität	deutsch	
Schule	1984 – 1988	Grundschule Langgöns und Katzenfurt
	1988 – 1997	Johanneum-Gymnasium Herborn
Zivildienst	1997 – 1998	Alten- und Pflegeheim der Arbeiterwohlfahrt in Herborn
Studium	1998 – 2002	Informatik an der Fachhochschule Darmstadt, Abschluss Diplom-Informatiker (FH)
	2003 – 2005	Erfüllung der Auflagen zur Annahme als Doktorand am Fachbereich Informatik der Technischen Universität Darmstadt
Promotionsvorbereitung	2005 – 2008	Anstellung als wissenschaftlicher Mitarbeiter am Fachgebiet Flugsysteme und Regelungstechnik der Technischen Universität Darmstadt
Beruf	2009 – zurzeit	Manager Product Development bei Lufthansa Systems FlightNav AG, Zürich, Schweiz